## 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

іі ОГЛАВЛЕНИЕ

7	Гру	ппы		19
	7.1	DFT-8	8	19
		7.1.1	Подробное описание	19
		7.1.2	Функции	19
			7.1.2.1 nmppsDFT8Fwd_32fcr()	19
	7.2	FFT-1	.6	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-3	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-6	34	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-1	.28	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-2	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-5	12	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-1	024	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-2	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-4	1096	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

iv ОГЛАВЛЕНИЕ

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	$nmppc Prod C \ \dots \dots$	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 \\ \dots \\ $	75
	7.33.1 Подробное описание	75
7.34	$nmppcFixMul32 \\ \ldots \\ $	76
	7.34.1 Подробное описание	76
7.35	$nmppcFixInv32 \ \dots $	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 \ \dots \ $	81
	7.39.1 Подробное описание	81
7.40	$nmppcFixDivMod 32 \ \dots $	82
	7.40.1 Подробное описание	82
7.41	nmppcFixSqrt64	83

ОГЛАВЛЕНИЕ vii

	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	Рункции 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	Інициализация и копирование	120
	7.71.1 Подробное описание	120
7.72	umppmCopyua	121
	7.72.1 Подробное описание	121
7.73	MTR_Copyau	122
	7.73.1 Подробное описание	122
7.74	MTR_Copy	123
	7.74.1 Подробное описание	123
7.75	umppmMul_mm	124
	7.75.1 Подробное описание	124
7.76	$\operatorname{mppmMul\_mv\_}$	126
	7.76.1 Подробное описание	126
7.77	$\operatorname{mppmMul\_mv\_\_AddC}$	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	${ m MTR\_SetVal}$	132
	7.82.1 Подробное описание	132

7.83	$MTR_{\perp}$	_Ge	tVal .						 			 	•	 					 133
	7.83.1	По	дробі	ное с	опис	ание			 			 		 					 133
7.84	Функц	ции	подд	ержі	КИ				 			 		 					 134
	7.84.1	По	дробі	ное с	опис	ание			 			 		 					 134
7.85	Вектор	рно	-матр	нРи	ые о	пера	ции		 			 	•	 		 			 135
	7.85.1	По	дробі	ное с	эпис	ание			 			 		 					 135
7.86	FFT-2	56							 			 		 					 136
	7.86.1	По	дробі	ное с	эпис	ание			 			 		 					 136
	7.86.2	Фу	ткци	и.					 			 		 					 136
		7.8	6.2.1	FF	T_F	$\sqrt{2}$	56()	•	 			 							 136
7.87	IFFT-2	256							 			 		 					 138
	7.87.1	По	дробі	ное с	опис	ание			 			 		 					 138
	7.87.2	Фу	'нкци	и.					 			 		 					 138
		7.8	7.2.1	FF	$T_{-}I$	nv25	6() .		 			 		 					 138
7.88	FFT-5	12							 			 		 					 141
	7.88.1	По	дробі	ное с	эпис	ание			 			 		 					 141
	7.88.2	Фу	'нкци	и.					 		•	 		 					 141
		7.8	8.2.1	FF	$T_F$	$7 \mathrm{wd} 5$	12()		 			 		 					 141
7.89	IFFT-5	512							 			 		 					 143
	7.89.1	По	дробі	ное с	эпис	ание			 		•	 		 					 143
	7.89.2	Фу	'нкци	И.					 			 		 					 143
		7.8	9.2.1	FF	$T_{\perp}I$	nv51	2() .		 			 		 					 143
7.90	FFT-10	024						•	 			 		 					 146
	7.90.1	По	дробі	ное с	эпис	ание			 			 		 					 146
	7.90.2	Фу	'нкци	.и					 			 		 					 146
		7.9	0.2.1	FF	$T_{-}F$	wd1	024()	) .	 			 		 					 146
7.91	IFFT-1	102	4						 			 		 					 148
	7.91.1	По	дробі	ное с	опис	ание			 			 		 					 148
	7.91.2	Фу	'нкци	и.					 			 		 					 148
		7.9	1.2.1	FF	T_I	nv10	24()		 			 		 					 148

ОГЛАВЛЕНИЕ хі

7.92 FFT-2048	51
7.92.1 Подробное описание	51
7.92.2 Функции	51
7.92.2.1 FFT_Fwd2048()	51
7.93 IFFT-2048	53
7.93.1 Подробное описание	53
7.93.2 Функции	53
7.93.2.1 FFT_Inv2048()	53
7.94 FFT-4096	56
7.94.1 Подробное описание	56
7.94.2 Функции	56
7.94.2.1 FFT_Fwd4096()	56
7.95 IFFT-4096	58
7.95.1 Подробное описание	58
7.95.2 Функции	58
7.95.2.1 FFT_Inv4096()	58
7.96 FFT-8192	30
7.96.1 Подробное описание	30
7.96.2 Функции	30
7.96.2.1 FFT_Fwd8192()	30
7.97 IFFT-8192	32
7.97.1 Подробное описание	32
7.97.2 Функции	32
7.97.2.1 FFT_Inv8192()	32
7.98 Свертка	34
7.98.1 Подробное описание	34
7.99 Масочная фильтрация	35
7.99.1 Подробное описание	35
7.100Изменение размеров	36
7.100.1 Подробное описание	36

7.101Быстрое преобразование Фурье
7.101.1 Подробное описание
7.101.2 Функции
7.101.2.1 nmppsFFTFree_32fcr()
7.102Элементарные математические функции
7.102.1 Подробное описание
7.103SIG_XCorr
7.103.1 Подробное описание
7.104nmppsSin_32f
7.104.1 Подробное описание
7.104.2 Функции
7.104.2.1 nmppsSin_32f()
7.105nmppsCos_32f
7.105.1 Подробное описание
7.105.2 Функции
7.105.2.1 nmppsCos_32f()
7.106nmppsDiv_32f
$7.106.1\Pi$ одробное описание
7.106.2 Функции
7.106.2.1 nmppsDiv_32f()
7.107nmppsExp_32f
7.107.1 Подробное описание
7.107.2 Функции
7.107.2.1 nmppsExp_32f()
7.108nmppsExp_64f
7.108.1 Подробное описание
7.108.2 Функции
7.108.2.1 nmppsExp_64f()
7.109nmppsLog_32f
7.109.1 Подробное описание

177
177
178
178
178
178
179
179
179
179
180
180
181
181
182
182
183
183
184
184
185
185
186
186
187
187
188
188
189
189

7.122SIG_SetResample
7.122.1 Подробное описание
7.123SIG_Resample_perf
7.123.1 Подробное описание
7.124Типы векторных данных
7.124.1 Подробное описание
7.124.2 Типы
7.124.2.1 nm1
7.124.2.2 nm16s
7.124.2.3 nm16s15b
7.124.2.4 nm16u
7.124.2.5 nm16u15b
7.124.2.6 nm2s
7.124.2.7 nm2u
7.124.2.8 nm32s
7.124.2.9 nm32s30b
7.124.2.10hm32s31b
7.124.2.11mm32u
7.124.2.12hm32u31b
7.124.2.13nm4s
7.124.2.14nm4u
7.124.2.15nm4u3b
7.124.2.16nm64s
7.124.2.17hm64s63b
7.124.2.18nm64u
7.124.2.19nm8s
7.124.2.20nm8s7b
7.124.2.21nm8u
7.124.2.22nm8u7b
7.124.2.23v16nm16s

7.124.2.24v16nm16u	. 201
7.124.2.25v16nm32s	. 201
7.124.2.26v16nm32u	. 202
7.124.2.27v16nm4b3u	. 202
7.124.2.28v16nm4u	. 202
7.124.2.29v16nm8s	. 202
7.124.2.30v16nm8s7b	. 202
7.124.2.3lv16nm8u	. 202
7.124.2.32v2nm32s	. 203
7.124.2.33·2nm32u	. 203
7.124.2.34v4nm16s	. 203
7.124.2.35v4nm16u	. 203
7.124.2.36v4nm32s	. 203
7.124.2.37v4nm32u	. 203
7.124.2.38v4nm8u	. 203
7.124.2.39·8nm16s	. 203
7.124.2.40/8nm16u	. 204
7.124.2.4lv8nm32s	. 204
7.124.2.42v8nm32u	. 204
7.124.2.43v8nm8s	. 204
7.124.2.44v8nm8u	. 204
7.125Типы скалярных данных	. 205
$7.125.1\Pi$ одробное описание	. 205
7.125.2 Типы	. 205
7.125.2.1 int15b	. 206
7.125.2.2 int16b	. 206
7.125.2.3 int1b	. 206
7.125.2.4 int2b	. 206
7.125.2.5 int30b	. 207
7.125.2.6 int31b	. 207

$7.125.2.7~\mathrm{int}$ 32b	207
7.125.2.8 int3b	207
7.125.2.9 int4b	208
7.125.2.10nt63b	208
7.125.2.1 lint 64b	208
7.125.2.12nt7b	208
7.125.2.13nt8b	209
7.125.2.14nint15b	209
7.125.2.15uint16b	209
7.125.2.16uint1b	209
7.125.2.17uint2b	210
7.125.2.18uint31b	210
7.125.2.19uint32b	210
7.125.2.20int3b	210
7.125.2.2 luint 4b	211
7.125.2.22uint 63b	211
7.125.2.23uint64b	211
7.125.2.24nint7b	211
7.125.2.25uint8b	211
7.126Функции поддержки	212
$7.126.1\Pi$ одробное описание	212
7.127Инициализация и копирование	213
$7.127.1\Pi$ одробное описание	213
7.128Арифметические операции	214
7.128.1 Подробное описание	214
7.129 Погические и бинарные операции	215
$7.129.1\Pi$ одробное описание	215
7.130Операции сравнения	216
7.130.1 Подробное описание	216
7.131Переупорядочивание и сортировка	217

ОГЛАВЛЕНИЕ хvii

7.131.1 Подробное описание	 217
7.132nmppsAbs	 218
$7.132.1\Pi$ одробное описание	 218
7.133nmppsAbs1	 219
7.133.1 Подробное описание	 219
7.134nmppsNeg	 220
$7.134.1\Pi$ одробное описание	 220
7.135nmppsAddC	 221
$7.135.1\Pi$ одробное описание	 221
7.136nmppsAdd	 222
$7.136.1\Pi$ одробное описание	 222
7.137nmppsAdd_AddC	 223
$7.137.1\Pi$ одробное описание	 223
7.138nmppsSubC	 224
7.138.1 Подробное описание	 224
7.139nmppsSubCRev	 225
7.139.1 Подробное описание	 225
7.140nmppsSub	 226
$7.140.1\Pi$ одробное описание	 226
7.141nmppsAbsDiff	 227
7.141.1 Подробное описание	 227
7.142nmppsAbsDiff1	 228
$7.142.1\Pi$ одробное описание	 228
7.143nmppsMulC	 229
$7.143.1\Pi$ одробное описание	 229
7.144nmppsMul_AddC	 230
7.144.1 Подробное описание	 230
7.145nmppsMulC_AddC	 231
$7.145.1\Pi$ одробное описание	 231
7.145.2 Функции	 231

$7.145.2.1 \text{ nmppsMulC\_AddC\_2x32s()} \dots$	
7.146nmppsRShiftC_MulC_AddC	
7.146.1 Подробное описание	
$7.147$ nmppsMulC_AddV_AddC	
7.147.1 Подробное описание	
7.148nmppsSumN	
7.148.1 Подробное описание	
7.149nmppsDivC	
7.149.1 Подробное описание	
7.150nmppsSum	
7.150.1 Подробное описание	
7.151nmppsDotProd	
7.151.1 Подробное описание	
7.152nmppsWeightedSum	
7.152.1 Подробное описание	
7.153nmppsNot	
7.153.1 Подробное описание	
7.154nmppsAndC	
7.154.1 Подробное описание	
7.155nmppsAnd	
7.155.1 Подробное описание	
7.156nmppsAnd4V	
7.156.1 Подробное описание	
7.157nmppsAndNotV	
7.157.1 Подробное описание	
7.158nmppsOrC	
7.158.1 Подробное описание	
7.159nmppsOr	
7.159.1 Подробное описание	
7.160nmppsOr3V	

7.160.1 Подробное описание	249
7.161nmppsOr4V	250
7.161.1 Подробное описание	250
7.162nmppsXorC	251
7.162.1 Подробное описание	251
7.163nmppsXor	252
7.163.1 Подробное описание	252
7.164nmppsMaskV	253
7.164.1 Подробное описание	253
7.165nmppsRShiftC	254
7.165.1 Подробное описание	254
7.166nmppsRShiftC	255
7.166.1 Подробное описание	255
7.167nmppsRShiftC_AddC	256
7.168nmppsDisplaceBits	257
7.168.1 Подробное описание	257
7.169nmppsSet-инициализация	259
7.169.1 Подробное описание	259
7.170nmppsRandUniform	260
7.170.1 Подробное описание	260
7.170.2 Функции	260
7.170.2.1 nmppsRandUniform_64s()	260
7.171nmppsRandUniform	261
7.171.1 Подробное описание	261
7.172nmppsRamp	262
7.172.1 Подробное описание	262
7.173nmppsConvert	263
7.173.1 Подробное описание	264
7.173.2 Функции	264
7.173.2.1 nmppsConvert_1s2s()	264

7.173.2.2 nmppsConvert_1u2u()
7.173.2.3 nmppsConvert_32s32fcr()
7.173.2.4 nmppsConvert_32sc32fcr()
7.173.2.5 nmppsConvert_32u32fcr()
7.173.2.6 nmppsConvertRisc_32u8u()
7.173.2.7 nmppsConvertRisc_8u32u()
7.173.2.8 nmppsJoin_32f()
7.174nmppsCopy
$7.174.1\Pi$ одробное описание
7.175nmppsCopyua
$7.175.1\Pi$ одробное описание
7.176nmppsSwap
$7.176.1\Pi$ одробное описание
7.177nmppsMax
$7.177.1\Pi$ одробное описание
7.177.2 Функции
7.177.2.1 nmppsMax_16s15b()
7.177.2.2 nmppsMax_32s31b()
7.177.2.3 nmppsMax_8s7b()
7.178nmppsMin
$7.178.1\Pi$ одробное описание
7.178.2 Функции
7.178.2.1 nmppsMin_16s15b()
7.178.2.2 nmppsMin_32s31b()
7.178.2.3 nmppsMin_8s7b()
7.179nmppsMaxIndx
$7.179.1\Pi$ одробное описание
7.180nmppsMinIndx
7.180.1 Подробное описание
7.181nmppsMinIndxVN

ОГЛАВЛЕНИЕ ххі

7.181.1 Подробное описание	279
7.182nmppsFirstZeroIndx	
7.182.1 Подробное описание	280
7.183nmppsFirstNonZeroIndx	281
7.183.1 Подробное описание	281
7.184nmppsLastZeroIndx	282
7.184.1 Подробное описание	282
7.185nmppsLastNonZeroIndx	283
7.185.1 Подробное описание	283
7.186nmppsMinEvery	284
7.186.1 Подробное описание	284
7.187nmppsMaxEvery	285
7.187.1 Подробное описание	285
7.188nmppsMinCmpLtV	287
7.188.1 Подробное описание	287
7.189nmppsCmpLt0	289
7.189.1 Подробное описание	289
7.190nmppsCmpEq0	290
7.190.1 Подробное описание	290
7.190.2 Функции	290
7.190.2.1 nmppsCmpEq0_16u15b()	291
7.190.2.2 nmppsCmpEq0_32u31b()	291
7.190.2.3 nmppsCmpEq0_8u7b()	291
7.191nmppsCmpMinMaxV	292
7.191.1 Подробное описание	292
7.192nmppsClipPowC	294
7.192.1 Подробное описание	294
7.193nmppsClipCC	295
7.193.1 Подробное описание	295
$7.194 nmpps Clip RS hift Convert\_Add C\_ \ \dots $	296

$7.194.1\Pi$ одробное описание
7.195nmppsClipConvert_AddC
7.195.1 Подробное описание
7.196nmppsCmpEqC
7.196.1 Подробное описание
7.197nmppsCmpNe0
7.197.1 Подробное описание
7.198nmppsCmpNeC
7.198.1 Подробное описание
7.199nmppsCmpEqV
7.199.1 Подробное описание
7.200nmppsCmpNeV
7.200.1 Подробное описание
7.201nmppsAddr
7.201.1 Подробное описание
7.202nmppsSetVal
7.202.1 Подробное описание
7.203nmppsGetVal
7.203.1 Подробное описание
7.204nmppsGetVal_(return)
7.204.1 Подробное описание
7.205VEC_QSort
7.205.1 Подробное описание
7.206nmppsRemap
7.206.1 Подробное описание
7.207nmppSplitTmp
7.208nmppSplit
7.209nmppMerge
7.210nmppSplit_32fcr
7.211nmppsDecimate

ОГЛАВЛЕНИЕ ххііі

7.212Типы данных	8
7.212.1 Подробное описание	8
7.213Векторные функции	9
7.213.1 Подробное описание	9
7.214Матричные функции	0
7.214.1 Подробное описание	0
$7.215 \Phi$ ункции обработки сигналов	1
7.215.1 Подробное описание	1
$7.216 \Phi$ ункции обработки изображений	2
7.216.1 Подробное описание	2
7.217Скалярные функции	3
$7.217.1\Pi$ одробное описание	3
7.217.1.1 Введение	3
7.218Базовые регистровые функции библиотеки	5
$7.218.1\Pi$ одробное описание	5
7.219контроль переполнения	6
$7.219.1\Pi$ одробное описание	6
7.219.2 Макросы	6
7.219.2.1 GetVec	6
7.219.3 Функции	7
7.219.3.1 operator<<()	7
7.220Элементарные функции	8
$7.220.1\Pi$ одробное описание	8
7.221функции взвешенного суммирования	9
$7.221.1\Pi$ одробное описание	9
7.222Целевые функции	0
$7.222.1\Pi$ одробное описание	0
7.223Vec_0_sub_data	1
$7.223.1\Pi$ одробное описание	1
7.224Vec_activate_data	2

7.224.1 Подробное описание
7.225 Vec_activate_data_add_0
7.225.1 Подробное описание
7.226Vec_activate_data_xor_data
7.226.1 Подробное описание
7.227Vec_activate_data_add_ram 338
7.227.1 Подробное описание
7.228Vec_Add_VV_shift
7.228.1 Подробное описание
7.229Vec_afifo
7.229.1 Подробное описание
7.230Vec_data
7.230.1 Подробное описание
7.231Vec_data_add_afifo
7.231.1 Подробное описание
7.232Vec_data_add_ram
7.232.1 Подробное описание
7.233Vec_data_and_ram
7.233.1 Подробное описание
7.234Vec_data_or_ram
7.234.1 Подробное описание
7.235Vec_data_sub_ram
7.235.1 Подробное описание
7.236Vec_data_xor_ram
7.236.1 Подробное описание
7.237Vec_FilterCoreRow2
7.237.1 Подробное описание
7.238Vec_FilterCoreRow4
7.238.1 Подробное описание
7.239Vec_FilterCoreRow8

7.239.1 Подробное описание
7.240Vec_And
$7.240.1\Pi$ одробное описание
7.241 Vec_Mask
7.241.1 Подробное описание
7.242Vec_Or
7.242.1 Подробное описание
7.243Vec_Xor
7.243.1 Подробное описание
7.244Vec_Abs
7.244.1 Подробное описание
7.245 Vec_Add
7.245.1 Подробное описание
7.246Vec_ClipExt
$7.246.1\Pi$ одробное описание
7.247Vec_ClipMul2D2W8_AddVr
$7.247.1\Pi$ одробное описание
7.248Vec_ClipMulNDNW2_AddVr
7.248.1 Подробное описание
7.249Vec_ClipMulNDNW4_AddVr
7.249.1 Подробное описание
7.250Vec_ClipMulNDNW8_AddVr
7.250.1 Подробное описание
7.251 Vec_IncNeg
7.251.1 Подробное описание
7.252Vec_Mul2D2W1_AddVr
7.252.1 Подробное описание
7.253Vec_Mul2D2W2_AddVr
7.253.1 Подробное описание
7.254Vec_Mul2D2W4_AddVr

7.254.1 Подробное описание
7.255Vec_Mul2D2W8_AddVr
7.255.1 Подробное описание
7.256Vec_Mul3D3W2_AddVr
$7.256.1\Pi$ одробное описание
7.257Vec_Mul3D3W8_AddVr
7.257.1 Подробное описание
7.258Vec_Mul4D4W2_AddVr
7.258.1 Подробное описание
7.259Vec_MulVN_AddVN
7.259.1 Подробное описание
7.260Vec_Sub
7.260.1 Подробное описание
7.261Vec_SubAbs
7.261.1 Подробное описание
7.262Vec_SubVN_Abs
7.262.1 Подробное описание
7.263Vec_Swap
7.263.1 Подробное описание
7.264Vec_MUL_2V4toW8_shift
7.264.1 Подробное описание
7.265 Vec_MUL_2V8toW16_shift
7.265.1 Подробное описание
7.266Vec_not_data
7.266.1 Подробное описание
7.267Vec_ram
7.267.1 Подробное описание
7.268Vec_ram_sub_data
7.268.1 Подробное описание
7.269Vec_vsum_activate_data_0

ОГЛАВЛЕНИЕ ххvii

$7.269.1\Pi$ одробное описание
7.270Vec_vsum_data_0
$7.270.1\Pi$ одробное описание
7.271Vec_vsum_data_afifo
$7.271.1\Pi$ одробное описание
7.272 Vec_vsum_data_vr
7.272.1 Подробное описание
7.273Vec_vsum_shift_data_0
7.273.1 Подробное описание
7.274 Vec_vsum_shift_data_vr
7.274.1 Подробное описание
7.275 Vec_vsum_shift_data_afifo
7.275.1 Подробное описание
7.276 Vec_CompareMinV
7.276.1 Подробное описание
7.277Vec_CompareMaxV
$7.277.1\Pi$ одробное описание
7.278 Vec_Dup ValueIn Vector8
7.278.1 Подробное описание
7.279Vec_DupValueInVector16
$7.279.1\Pi$ одробное описание
7.280Vec_BuildDiagWeights8
7.280.1 Подробное описание
7.281Vec_BuildDiagWeights16
7.281.1 Подробное описание
7.282 Vec_MaxVal_v8nm8s
$7.282.1\Pi$ одробное описание
7.283Vec_MaxVal_v4nm16s
7.283.1 Подробное описание
7.284Vec_MaxVal
7.284.1 Подробное описание
7.285 Vec_MinVal_v8nm8s
$7.285.1\Pi$ одробное описание
7.286Vec_MinVal_v4nm16s
$7.286.1\Pi$ одробное описание
7.287Vec_MinVal
$7.287.1\Pi$ одробное описание
7.288Vec_AccMul1D1W32_AddVr
7.288.1 Подробное описание

3	Кла	ссы	397
	8.1	Класс C_2DSubPixelMinPosition	397
		8.1.1 Подробное описание	397
	8.2	Kласс C_2DTrigSubPixelMinPosition	397
		8.2.1 Подробное описание	398
	8.3	Класс C_Allocator32	398
		8.3.1 Подробное описание	398
	8.4	Шаблон класса C_BoxImg< T >	399
		8.4.1 Подробное описание	399
	8.5	Шаблон класса C_BoxVec< T >	399
		8.5.1 Подробное описание	400
	8.6	Класс С_Неар	400
		8.6.1 Подробное описание	401
		8.6.2 Методы	401
		8.6.2.1 AllocateMaxAvail()	401
	8.7	Шаблон класса С_Img< Т >	401
		8.7.1 Подробное описание	402
	8.8	Класс С_MultiHeap	402
		8.8.1 Подробное описание	403
	8.9	Класс C_PlessyCornerDetector	403
		8.9.1 Подробное описание	404
	8.10	Класс C_PlessyCornerDetector_16s	404
		8.10.1 Подробное описание	405
	8.11	Класс C_PlessyCornerDetector_32f	405
		8.11.1 Подробное описание	406
	8.12	Шаблон класса C_RingBufferRemote< T >	406
		8.12.1 Подробное описание	407
	8.13	Шаблон класса С_WarpImg< Т >	407
		8.13.1 Подробное описание	408
	8.14	Шаблон класса CIMG_FIR< nmbits_in, nmbits_out >	408

ОГЛАВЛЕНИЕ ххіх

	8.14.1 Подробное описание	. 409
	8.14.2 Конструктор(ы)	. 409
	8.14.2.1 CIMG_FIR()	. 409
	8.14.3 Методы	. 410
	8.14.3.1 Filter()	. 410
	8.14.3.2 SetWeights()	. 410
8.15	Структура ds_struct	. 411
	8.15.1 Подробное описание	. 411
8.16	Класс EnterHardMode	. 411
	8.16.1 Подробное описание	. 411
8.17	Класс I_2DSubPixelMinPosition	. 411
	8.17.1 Подробное описание	. 412
8.18	Kласс I_PlessyCornerDetector	. 412
	8.18.1 Подробное описание	. 412
8.19	Структура int15in16x4	. 412
	8.19.1 Подробное описание	. 413
8.20	Структура int30in32x2	. 413
	8.20.1 Подробное описание	. 413
8.21	Структура int31in32x2	. 413
	8.21.1 Подробное описание	. 413
8.22	Шаблон класса mtr< T >	. 414
	8.22.1 Подробное описание	. 415
8.23	Структура nm16sc	. 415
	8.23.1 Подробное описание	. 415
8.24	Класс nmchar	. 416
	8.24.1 Подробное описание	. 416
8.25	Шаблон класса nmchar1D< N >	. 416
	8.25.1 Подробное описание	. 416
8.26	Шаблон класса nmchar2D< Y, X >	. 417
	8.26.1 Подробное описание	. 417

8.27	Шаблон класса nmintpack $<$ T $>$	417
	8.27.1 Подробное описание	417
8.28	Шаблон класса nmmtr< T >	418
	8.28.1 Подробное описание	419
8.29	Структура NmppiFFTSpec_32fcr	419
	8.29.1 Подробное описание	419
8.30	Структура NmppsFFTSpec	419
	8.30.1 Подробное описание	420
8.31	Структура NmppsFFTSpec_32fcr	420
	8.31.1 Подробное описание	420
8.32	Структура NmppsFrame_16s	420
	8.32.1 Подробное описание	420
8.33	Структура NmppsFrame_16u	420
	8.33.1 Подробное описание	421
8.34	Структура NmppsFrame_32s	421
	8.34.1 Подробное описание	421
8.35	Структура NmppsFrame_32u	421
	8.35.1 Подробное описание	421
8.36	Структура NmppsFrame_64s	421
	8.36.1 Подробное описание	422
8.37	Структура NmppsFrame_64u	422
	8.37.1 Подробное описание	422
8.38	Структура NmppsFrame_8s	422
	8.38.1 Подробное описание	422
8.39	Структура NmppsFrame_8u	422
	8.39.1 Подробное описание	423
8.40	Структура NmppsMallocSpec	423
	8.40.1 Подробное описание	423
8.41	Структура NmppsTmpSpec	423
	8.41.1 Подробное описание	423

8.42	Структура nmreg	424
	8.42.1 Подробное описание	424
8.43	Kласс nmshort	424
	8.43.1 Подробное описание	424
8.44	Шаблон класса nmshort2D < Y, X >	425
	8.44.1 Подробное описание	425
8.45	Шаблон класса nmvecpack < T >	425
	8.45.1 Подробное описание	426
8.46	Структура RGB32_nm10s	427
	8.46.1 Подробное описание	427
8.47	Структура RGB32_nm10u	427
	8.47.1 Подробное описание	427
8.48	Структура RGB32_nm8s	427
	8.48.1 Подробное описание	428
8.49	Структура RGB32_nm8u	428
	8.49.1 Подробное описание	428
8.50	Структура RGB64_nm16u	428
	8.50.1 Подробное описание	428
8.51	Kласс RPoint	429
	8.51.1 Подробное описание	429
8.52	Структура S_BufferInfo	429
	8.52.1 Подробное описание	430
8.53	Структура S_IMG_FilterKernel	430
	8.53.1 Подробное описание	430
8.54	Структура S_IMG_FilterKernel_32s32s	430
	8.54.1 Подробное описание	430
8.55	Структура s_int32x2	431
	8.55.1 Подробное описание	431
8.56	Структура s_nm32fc	431
	8.56.1 Подробное описание	431

8.57	Структура s_nm32fcr	431
	8.57.1 Подробное описание	431
8.58	Структура s_nm32sc	432
	8.58.1 Подробное описание	432
8.59	Структура s_nm64sc	432
	8.59.1 Подробное описание	432
	8.59.2 Данные класса	432
	8.59.2.1 im	432
	8.59.2.2 re	433
8.60	Структура s_v16nm16s	433
	8.60.1 Подробное описание	433
8.61	Структура s_v16nm16u	433
	8.61.1 Подробное описание	433
8.62	Структура s_v16nm32s	434
	8.62.1 Подробное описание	434
8.63	Структура s_v16nm32u	434
	8.63.1 Подробное описание	434
8.64	Структура s_v16nm4u	434
	8.64.1 Подробное описание	435
8.65	Структура s_v16nm8s	435
	8.65.1 Подробное описание	435
8.66	Структура s_v16nm8u	435
	8.66.1 Подробное описание	435
8.67	Структура s_v2nm32s	436
	8.67.1 Подробное описание	436
8.68	Структура s_v2nm32u	436
	8.68.1 Подробное описание	436
8.69	Структура s_v4nm16s	436
	8.69.1 Подробное описание	437
8.70	Структура s_v4nm16u	437

ОГЛАВЛЕНИЕ хххііі

	8.70.1 Подробное описание	437
8.71	Структура s_v4nm32s	437
	8.71.1 Подробное описание	437
8.72	Структура s_v4nm32u	438
	8.72.1 Подробное описание	438
8.73	Структура s_v4nm8u	438
	8.73.1 Подробное описание	438
8.74	Структура s_v8nm16s	438
	8.74.1 Подробное описание	439
8.75	Структура s_v8nm16u	439
	8.75.1 Подробное описание	439
8.76	Структура s_v8nm32s	439
	8.76.1 Подробное описание	439
8.77	Структура s_v8nm32u	440
	8.77.1 Подробное описание	440
8.78	Структура s_v8nm8s	440
	8.78.1 Подробное описание	440
8.79	Структура s_v8nm8u	440
	8.79.1 Подробное описание	441
8.80	Структура SpecTmp1	441
	8.80.1 Подробное описание	441
8.81	Структура spot_struct	441
	8.81.1 Подробное описание	442
8.82	Структура tagSegmentInfo	442
	8.82.1 Подробное описание	442
8.83	Шаблон класса tfixpoint $<$ T, point $>$	442
	8.83.1 Подробное описание	443
8.84	Структура Tmp2BuffSpec	443
	8.84.1 Подробное описание	444
8.85	Kласс uint16ptr	444
	8.85.1 Подробное описание	444
8.86	Kлаcc uint8ptr	445
	8.86.1 Подробное описание	445
8.87	Структура v16nm4s	445
	8.87.1 Подробное описание	446
8.88	Структура v4nm8s	446
	8.88.1 Подробное описание	446
8.89	Шаблон класса vec< T $>$	446
	8.89.1 Подробное описание	448

9 Файлы							
	9.1	Файл	D:/GIT/	nmpp/include/nmblas.h		449	
		9.1.1	Подробі	ное описание		450	
	9.2	2 Файл D:/GIT/nmpp/include/nmtype.h					
		9.2.1	Подробі	ное описание		453	
		9.2.2	Макрос	ы		454	
			9.2.2.1	CAPACITY_nm64s		454	
			9.2.2.2	VEC_NM16S		454	
			9.2.2.3	VEC_NM16U		454	
			9.2.2.4	VEC_NM32S		455	
			9.2.2.5	VEC_NM4U		455	
			9.2.2.6	VEC_NM8S		455	
<b>A</b> J	гфави	итный ј	указателі	Ь		457	

# Введение

1.1 Introduction

This is the introduction.

- 1.2 Installation
- 1.2.1 Step 1: Opening the box

 ${\rm etc...}$ 

2 Введение

# Ошибки

Файл nmblas.h

# Алфавитный указатель групп

# 3.1 Группы

# Полный список групп.

BLASS-LEVEL1
FFT-256
IFFT-256
FFT-512
IFFT-512
FFT-1024
IFFT-1024
FFT-2048
IFFT-2048
FFT-4096
IFFT-4096
FFT-8192
IFFT-8192
Типы данных
Типы векторных данных
Типы скалярных данных
Векторные функции
n 100
Элементарные математические функции
nmppsSin_32f
nmppsSin_32f
nmppsSin_32f
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
nmppsSin_32f172nmppsCos_32f173nmppsDiv_32f174nmppsExp_32f175nmppsExp_64f176nmppsLog_32f177nmppsLog_64f178nmppsPowx_64f179Функции поддержки212
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64         nmppsFree       65
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64         nmppsFree       65         nmppsAddr_       306
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64         nmppsFree       65         nmppsAddr_       306         nmppsSetVal_       307
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64         nmppsFree       65         nmppsAddr_       306         nmppsSetVal_       307         nmppsGetVal       308
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64         nmppsFree       65         nmppsAddr_       306         nmppsSetVal_       307         nmppsGetVal_       308         nmppsGetVal_(return)       309
nmppsSin_32f       172         nmppsCos_32f       173         nmppsDiv_32f       174         nmppsExp_32f       175         nmppsExp_64f       176         nmppsLog_32f       177         nmppsLog_64f       178         nmppsPowx_64f       179         Функции поддержки       212         nmppsMalloc       64         nmppsFree       65         nmppsAddr_       306         nmppsSetVal_       307         nmppsGetVal       308

nmppsRandUniform											
$\operatorname{nmppsRandUniform}_{-} \ldots$											
nmppsRamp											
nmppsConvert											
nmppsCopy											
nmppsCopyua											
nmppsSwap											
Арифметические операции											
$\operatorname{nmppsAbs}$											
${ m nmppsAbs1}$											
$\mathrm{nmppsNeg}$											
$\operatorname{nmppsAddC}$											
${ m nmppsAdd}$											
$\operatorname{nmppsAdd}_{-}\operatorname{AddC}$											
$\operatorname{nmppsSubC}$											
${ m nmppsSubCRev}$											
$\mathrm{nmppsSub}$											
${ m nmppsAbsDiff}$											
$\mathrm{nmppsAbsDiff1}$											
$\operatorname{nmppsMulC}$											
$nmppsMul\_AddC$		 	 	 			 				. 230
$nmppsMulC\_AddC$		 	 	 			 				. 231
$nmppsRShiftC\_MulC\_AddC$	١	 	 	 			 				. 233
$nmppsMulC \overline{AddV} \overline{AddC}$		 	 	 			 				. 234
m nmppsSumN		 	 	 			 				. 235
$\operatorname{nmppsDivC}$		 	 	 			 				. 236
nmppsSum		 	 	 			 				. 238
nmppsDotProd		 	 	 			 				. 239
$nmppsWeightedSum \dots$		 	 	 			 				. 241
Логические и бинарные операц	ии .	 	 	 			 				215
m nmppsNot											
$\operatorname{nmppsAndC}$											
$\operatorname{nmppsAnd}$											
ho = 1000  ho $ ho = 1000  ho$ $ ho = 1000  ho$											
$\operatorname{nmppsAndNotV}$											
nmppsOrC											
nmppsOr											
$ m nmppsOr3V\_$											
mmppsoro =											
nmppsOr4V			 	 			 				249
nmppsOr4V		 	 	 			 				249 250
$\operatorname{nmppsXorC}^{-}$		 	 	 			 	   	 •		. 249 . 250 . 251
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	 	 	 	 	  	  	     		 	 	. 249 . 250 . 251 . 252
nmppsXorC		 · · · · · · · · · · · · · · · · · · ·	 	 			   	 	 	 	. 249 . 250 . 251 . 252 . 253
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		 	 	 	· · · · · · · · · · · · · · · · · · ·		 		 	 	. 249 . 250 . 251 . 252 . 253 . 254
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		 			<ul><li>.</li><li>.</li><li>.</li><li>.</li><li>.</li><li>.</li><li>.</li><li>.</li><li>.</li><li>.</li></ul>		 			 	. 249 . 250 . 251 . 252 . 253 . 254 . 255
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		 					 		 		. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		 				· · · · · · · · · · · · · · · · · · ·	 				. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257
nmppsXorC		 					 · · · · · · · · · · · · · · · · · · ·				249 250 251 252 253 254 255 256 257
nmppsXorC nmppsXor nmppsXor nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsDisplaceBits							 				. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 216
nmppsXorC nmppsXor nmppsMaskV nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC nddC nmppsDisplaceBits Операции сравнения nmppsMax nmppsMin											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 216 . 271 . 273
nmppsXorC nmppsXor nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsDisplaceBits пmppsMax nmppsMax nmppsMin nmppsMaxIndx nmppsMaxIndx											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 216 . 271 . 273 . 275
nmppsXorC nmppsXor nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsDisplaceBits											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 216 . 271 . 273 . 275 . 277
nmppsXorC											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 . 216 . 271 . 273 . 275 . 277 . 279
nmppsXorC nmppsXorC nmppsMaskV nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsDisplaceBits Oперации сравнения nmppsMax nmppsMin nmppsMaxIndx nmppsMinIndx nmppsMinIndxVN nmppsFirstZeroIndx											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 . 271 . 273 . 275 . 279 . 280
nmppsXorC nmppsXorC nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsDisplaceBits Oперации сравнения nmppsMax nmppsMin nmppsMinIndx nmppsMinIndxVN nmppsFirstZeroIndx nmppsFirstNonZeroIndx nmppsFirstNonZeroIndx											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 . 271 . 273 . 275 . 279 . 280 . 281
nmppsXorC nmppsXorC nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC AddC nmppsDisplaceBits											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 . 271 . 273 . 275 . 279 . 280 . 281
nmppsXorC nmppsXorC nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC AddC nmppsDisplaceBits Операции сравнения nmppsMax nmppsMin nmppsMin nmppsMinIndx nmppsMinIndx nmppsFirstZeroIndx nmppsFirstNonZeroIndx nmppsLastZeroIndx nmppsLastNonZeroIndx nmppsLastNonZeroIndx nmppsLastNonZeroIndx nmppsLastNonZeroIndx nmppsLastNonZeroIndx nmppsLastNonZeroIndx											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 . 271 . 273 . 275 . 280 . 281 . 282 . 283
nmppsXorC nmppsXorC nmppsMaskV nmppsRShiftC nmppsRShiftC nmppsRShiftC nmppsRShiftC AddC nmppsDisplaceBits											. 249 . 250 . 251 . 252 . 253 . 254 . 255 . 256 . 257 . 273 . 275 . 277 . 279 . 280 . 281 . 282 . 283 . 284

3.1 Группы 7

$\operatorname{nmppsMinCmpLtV}$	
nmppsCmpLt0	
nmppsСmpьqo nmppsCmpMinMaxV	
nmppsClipPowC	
$rac{1}{1} rac{1} rrac{1}{1} rac{1}{1} $	
$nmppsClipRShiftConvert\_AddC\_\dots\dots$	
$nmppsClipConvert\_AddC\_\dots\dots\dots$	
nmppsCmpEqC	
nmppsCmpNe0	
nmppsCmpFeV	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
Переупорядочивание и сортировка	
VEC QSort	
ho = 0 $ ho = 0$ $ ho$	
$\operatorname{nmppSplitTmp}$	
$\operatorname{nmppSplit}$	
nmppMerge	
nmppSplit_32fcr	
nmppsDecimate	
Матричные функции	
Инициализация и копирование	
mmppmcopyua	
MTR Copy	
Функции поддержки	
MTR Malloc	
MTR_Free	
$\mathrm{MTR}_{-}\mathrm{Addr}\dots\dots\dots\dots\dots\dots\dots$	
MTR_SetVal	
MTR_GetVal	
Векторно-матричные операции	
nmppmMul_mm	
nmppmMul_mvAddC	127
MTR_ProdUnitV	
– Функции обработки сигналов	
Свертка	
SIG_XCorr	
Масочная фильтрация	$. \dots \dots$
$SIG\_Median3$	
КИХ-фильтрация	
<del>-</del>	
=	
Изменение размеров	
SIG ResampleDown2	
$SIG\_ResampleUp3Down2$	
SIG_CreateResample	
SIG_SetResample	
SIG_Resample_perf	
Быстрое преобразование Фурье	
FFT-16	

$ ext{FFT-32}$	$\sim 23$
FFT-64	2
FFT-128	
FFT-256	29
FFT-512	3
FFT-1024	33
FFT-2048	3
FFT-4096	3'
IDFT-8	39
IFFT-16	40
IFFT-32	4:
IFFT-64	4
IFFT-128	40
IFFT-256	48
IFFT-512	50
IFFT-1024	55
IFFT-2048	54
IFFT-4096	50
FFT-Common	58
IFFT-Common	6
Функции обработки изображений	325
Floodfill	
Переупорядочивание изображений	
Блочное переупорядочивание	
IMG_SplitIntoBlocks	
IMG_MergeFromBlocks	
Арифметические действия	
Масочная фильтрация	
КИХ-фильтрация	
Инициализация и копирование	
IMG_Convert	100
IMG_RGB32ToGray	10'
Функции поддержки	11'
IMG Free	115
IMG Release	113
– Функции графического вывода текста	118
Скалярные функции	32
Инициализация	
m nmppsRand	
Integer operations	
nmppcSqrt	
Fix-point 64	
$\mathrm{nmppcFixSqrt}64$	
nmppcDoubleToFix64	
nmppcFix64ToDouble	
$\mathrm{nmppcFixDiv}64\ldots\ldots\ldots\ldots\ldots\ldots$	
$\operatorname{nmppcFixSinCos}64\ldots\ldots\ldots\ldots\ldots\ldots\ldots$	
${ m nmppcFixArcTan 64}$	88
Fix-point 32	9
nmppcFixExp32	70
$\operatorname{nmppcFixSinCos}32\ldots\ldots\ldots\ldots\ldots\ldots\ldots$	7
nmppcFixArcTan32	75
${ m nmppc}{ m Double}{ m ToFix}32$	7
${ m nmppcFix}32{ m ToDouble}$	74
$\operatorname{nmppcFixSqrt}32$	7
nmppcFixMul32	70
nmppcFixInv32	7'

3.1 Группы 9

nmppcTblFixArcSin32	
nmppcTblFixArcCos32	
$\operatorname{nmppcTblFixCos}32$	
$\operatorname{nmppcTblFixSin32}$	
nmppcFixDivMod32	
$\operatorname{nmppcFix} 64 \operatorname{Exp} 01$	89
Арифметические операции	96
nmppcDivC	68
nmppcProdC	69
Функции деинтерлейсинга	97
Базовые регистровые функции библиотеки	325
Элементарные функции	
Vec 0 sub data	
Vec activate data	
Vec activate data add 0	
Vec activate data xor data	
Vec activate data add ram	
Vec afifo	
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	
функции взвешенного суммирования	
$Vec\_ClipMul2D2W8\_AddVr \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	
$Vec\_ClipMulNDNW2\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_ClipMulNDNW4\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_ClipMulNDNW8\_AddVr  .  .  .  .  .  .  .  .  .  $	
${ m Vec\_Mul2D2W1\_AddVr}$	360
$Vec\_Mul2D2W2\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_Mul2D2W4\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_Mul2D2W8\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_Mul3D3W2\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_Mul3D3W8\_AddVr  .  .  .  .  .  .  .  .  .  $	
$Vec\_Mul4D4W2\_AddVr  .  .  .  .  .  .  .  .  .  $	366
Vec_MulVN_AddVN	
$Vec\_vsum\_data\_0  .  .  .  .  .  .  .  .  .  $	
Vec_vsum_data_vr	380
$Vec\_vsum\_shift\_data\_0\ \dots$	381
Vec_vsum_shift_data_vr	
Vec_vsum_shift_data_afifo	383
 Целевые функции	
Vec Add VV shift	
Vec data add afifo	
Vec FilterCoreRow2	
Vec FilterCoreRow4	
Vec FilterCoreRow8	

$Vec Abs \dots \dots$			 										. 3	352
Vec ClipExt			 										. 3	354
$\operatorname{Vec} \operatorname{\underline{IncNeg}} \ldots \ldots$			 										. 3	359
$\operatorname{Vec}^{-}\operatorname{SubAbs}$			 										. 3	369
Vec SubVN Abs			 										. 3	370
$\operatorname{Vec}^{-}\operatorname{Swap}$			 										. 3	371
Vec MUL 2V4toW8 shi	ift		 										. 3	372
Vec MUL 2V8toW16 sl	nift .		 										. 3	373
$\operatorname{Vec\_vsum\_data\_afifo}$			 										. 3	379
$Vec\_CompareMinV$			 										. 3	384
Vec CompareMaxV			 										. 3	385
${ m Vec}^{-}_{ m DupValueInVector8}$			 										. 3	386
$Vec\_DupValueInVector16$			 										. 3	387
$Vec\_BuildDiagWeights8$ .			 										. 3	388
Vec_BuildDiagWeights16			 										. 3	389
$Vec\_MaxVal\_v8nm8s$			 										. 3	<mark>39</mark> 0
$Vec\_MaxVal\_v4nm16s$ .			 										. 3	391
$Vec\_MaxVal \dots \dots$			 										. 3	392
$Vec\ MinVal\ v8nm8s$			 										. 3	393
Vec MinVal v4nm16s .			 										. 3	394
$\operatorname{Vec}^{-}\operatorname{MinVal}^{-}\ldots\ldots$			 										. 3	395
${ m Vec}_{ m AccMul1D1W32}_{ m Ad}$	dVr .		 										. 3	396
контроль переполнения		 										 	9	326

# Иерархический список классов

# 4.1 Иерархия классов

# Иерархия классов.

C_Allocator32
C MultiHeap
C BoxImg< T >
$C  BoxVec < T > \dots \dots$
C_Heap
C Img $<$ T $>$
$C_RingBufferRemote < T > \dots 406$
C_WarpImg< T >
CIMG_FIR< nmbits_in, nmbits_out >
ds_struct
EnterHardMode
I_2DSubPixelMinPosition
$C_2DSubPixelMinPosition$
$C\_2DTrigSubPixelMinPosition \dots 397$
I_PlessyCornerDetector
C_PlessyCornerDetector
C PlessyCornerDetector 16s
C_PlessyCornerDetector_32f
$\operatorname{int}15 \operatorname{in}16 \operatorname{x}4$
int 30 in 32 x 2
int31in32x2
mtr< T >
nm16sc
nmchar
nmchar1D< N >
nmchar2D< Y, X >
nmintpack< T >
nmmtr< T >
NmppiFFTSpec_32fcr
NmppsFFTSpec
NmppsFFTSpec_32fcr
NmppsFrame_16s
NmppsFrame_16u
NmppsFrame 32s

NmppsFrame_32u	 421
NmppsFrame_64s	 421
$NmppsFrame\_64u \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	 422
NmppsFrame_8s	 422
NmppsFrame_8u	 422
$NmppsMallocSpec \ \dots $	 423
NmppsTmpSpec	 423
nmreg	 424
nmshort	 424
nmshort2D< Y, X >	 425
$nmvecpack < T > \dots \dots$	 425
RGB32 nm10s	 427
RGB32 nm10u	 427
RGB32 nm8s	 427
RGB32 nm8u	
RGB64 nm16u	
RPoint	
S BufferInfo	
S IMG FilterKernel	
S IMG FilterKernel 32s32s	
s int32x2	
s nm32fc	
s nm32fcr	
s nm32sc	
s nm64sc	
s v16nm16s	
s_v16nm16u	
s_v16nm32s	
s_v16nm32u	
s_v16nm4u	
s_v16nm8s	
s_v16nm8u	
s_v10mm3us	
s_v2nm32u	
s_v2nm32d	
s_v4nm16u	
s_v4nm32s	
s_v4nm32u	 437
s_v4nm8u	
s_v8nm16u	
s_v8nm32s	
s_v8nm32u	
s_v8nm8s	
s_v8nm8u	
SpecTmp1	
spot_struct	
tagSegmentInfo	
tfixpoint < T, point >	
Tmp2BuffSpec	
uint16ptr	
uint8ptr	
v16nm4s	
v4nm8s	
$\operatorname{vec} < T > \dots$	 446

# Алфавитный указатель классов

# 5.1 Классы

### Классы с их кратким описанием.

NmppsFrame_32u									 	42	11
$NmppsFrame\_64s$									 	42	!1
NmppsFrame 64u									 	42	$^{2}$
NmppsFrame 8s									 	42	22
											22
= = =											
nmreg											
•											
nmshort											
· · · · · · · · · · · · · · · · · · ·											
1											
RGB32_nm10s									 	42	!7
RGB32_nm10u									 	42	!7
RGB32 nm8s									 	42	<u>?</u> 7
RGB32 nm8u									 	42	18
RGB64 nm16u									 	42	18
RPoint											
S BufferInfo									 		
класс буфер -	ропоп	DOK D	попо	о вил	тамо	й попол	uunekon	помат		42	'n
0 1 1							и ческои				
$S_{IMG}_{FilterKernel}$	=										
<del>-</del>										43	
<del>-</del>									 	43	
s_nm32fcr									 	43	1
$s_nm32sc$									 	43	32
$s nm64sc \dots$									 	43	32
s v16nm16s									 	43	33
s v16nm16u									 	43	33
s v16nm32s									 	43	34
<del>-</del>										43	
<del>-</del>										43	
<del>-</del>										43	
<del>-</del>										43	
<del>-</del>										43	
<del>-</del>										43	
s_v2nm32u s_v4nm16s											
<del></del>										43	
s_v4nm16u										43	
s_v4nm32s									 	43	
<del>-</del>									 		
s_v4nm8u									 	43	-
s_v8nm16s									 	43	8
$s_v8nm16u$									 	43	39
$s_v8nm32s$									 	43	39
s v8nm32u									 	44	10
s v8nm8s									 	44	10
s v8nm8u									 	44	10
$\operatorname{SpecTmp1}$									 	44	11
spot struct										44	
. T. C										44	
0 0										44	
tfixpoint < T, point >									 		
Tmp2BuffSpec									 	. 44	
uint16ptr									 	44	
uint8ptr									 	44	
v16nm4s									 	44	
v4nm8s									 	44	16
$vec < T > \dots$									 	44	6

# Список файлов

# 6.1 Файлы

Полный список документированных файлов.

	' '	m lude/crtdbg2.h										??
		m lude/fft.h										??
D:/	$/\mathrm{GIT/nmpp/inc}$	$ m lude/fft\_32fcr.h$						 	 	 		??
D:/	$/\mathrm{GIT/nmpp/inc}$	lude/fftexp.h						 	 	 		??
D:/	$/\mathrm{GIT/nmpp/inc}$	${ m lude/macros\_fpt}$	ı.h					 	 	 		??
D:/	$/\mathrm{GIT/nmpp/inc}$	lude/malloc32.h						 	 	 		??
D:/	/GIT/nmpp/inc	lude/metric.h .						 	 	 		??
D:/	/GIT/nmpp/inc	lude/minrep.h .						 	 	 		??
D:/	/GIT/nmpp/inc	${ m lude/multiheap.h}$	1					 	 	 		??
D:/	/GIT/nmpp/inc	lude/nmblas.h										
	Nmblas (N	VeuroMatrix Basi	c Linear	Algebr	a Sub	routin	es)	 	 	 	. 4	49
D:/	/GIT/nmpp/inc	${ m lude/nmchar.h}$						 	 	 		??
D:/	$/\mathrm{GIT/nmpp/inc}$	lude/nmdef.h .						 	 	 		??
D:	/GIT/nmpp/inc	${ m lude/nmplc.h}$						 	 	 		??
		lude/nmpli.h .										??
		lude/nmplm.h .										??
		lude/nmpls.h .										??
D:/	$/\mathrm{GIT/nmpp/inc}$	lude/nmplv.h .						 	 	 		??
D:/	$/\mathrm{GIT/nmpp/inc}$	${ m lude/nmpp.h}$ .						 	 	 		??
D:	/GIT/nmpp/inc	${ m lude/nmshort.h}$						 	 	 		??
D:	/GIT/nmpp/inc	${ m lude/nmtl.h}$						 	 	 		??
		lude/nmtype.h										51
D:	/GIT/nmpp/inc	lude/ringremote.	h					 	 	 		??
D:	/GIT/nmpp/inc	lude/test.h						 	 	 		??
D:	/GIT/nmpp/inc	${ m lude/tfixpoint.h}$						 	 	 		??
D:	/GIT/nmpp/inc	m lude/nmplc/cAri	${ m thmetic.h}$					 	 	 		??
D:	/GIT/nmpp/inc	m lude/nmplc/cfixp	$\mathrm{nt}32.\mathrm{h}$					 	 	 		??
		m lude/nmplc/cfixp										??
D:	/GIT/nmpp/inc	m lude/nmplc/cInit	.h					 	 	 		??
		m lude/nmplc/cInter										??
D:	/GIT/nmpp/inc	${ m lude/nmplc/nmp}$	lc.h					 	 	 		??
D:	/GIT/nmpp/inc	m lude/nmpli/filter	.h					 	 	 		??
		${ m lude/nmpli/iArit}$										??
		lude/nmpli/iCell										??
		lude/nmpli/iDef.										??

Список файлов

	/ PP/ / P/	??
	, , , , , , , , , , , , , , , , , , , ,	??
D:	/GIT/nmpp/include/nmpli/iFiltration.h	??
D:	/GIT/nmpp/include/nmpli/iFloodFill.h	??
		??
		??
	, , 11, , 1, 0	??
		· · ??
	, , <b>11</b> , , <b>1</b> ,	??
	, , 11, , 1 , 9	: : ??
	, , , , , , , , , , , , , , , , , , , ,	
	/ / ±±/ / ± /	??
	, , , , , , , , , , , , , , , , , , , ,	??
	, , 11, , 1 , 1	??
	, , , , , , , , , , , , , , , , , , , ,	??
		??
D:	$/\mathrm{GIT/nmpp/include/nmpli/warpimg.h}$	??
D:	$/\mathrm{GIT/nmpp/include/nmplm/mInit.h}$	??
D:	$/\mathrm{GIT/nmpp/include/nmplm/mInverse.h}$	??
		??
		??
	, , , , , , , , , , , , , , , , , , , ,	??
D.		· · ??
		??
		??
	11     1   _	
	, , , , , , , , , , , , , , , , , , , ,	??
	, , , , , , , , , , , , , , , , , , , ,	??
	, , , , , , , , , , , , , , , , , , , ,	??
	, , , , , , , , , , , , , , , , , , , ,	??
	/ / ±±/ / ± /	??
	/ / ±±/ / ± /	??
	, , 11, , 1	??
D:	, , 11, , 1	??
D:	$/\mathrm{GIT/nmpp/include/nmplv/nmtl.h}$	??
D:	/GIT/nmpp/include/nmplv/vArithmetics.h	??
		??
		??
	/ / II/ / I	??
	/ FI/ / F /	??
		??
		??
	, , <u>, , , , , , , , , , , , , , , , , </u>	??
	, , , , , , , , , , , , , , , , , , , ,	:: ??
		??
		??
	1 1 1 1 1	??
		??
		??
D:		??
D:	$/\mathrm{GIT/nmpp/include/nmtl/tmatrix.h}$	??
		??
		??
		??
		??
		??
		??
		: : ??
		??
υ:	/GIT/nmpp/include/rpc/aura-exports.c	??

 $6.1 \Phi$ айлы 17

$D:/GIT/nmpp/include/rpc/rpc-host.h \\ \ldots \\ $	??
D:/GIT/nmpp/include/rpc/rpc-nmc-func.h	??
D: /GIT/nmpn/include/rpc/rpc-nmc h	??

18 Список файлов

# Группы

# 7.1 DFT-8

Функция для вычисления прямого ДП $\Phi$  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 ДП $\Phi$  с плавающей точкой над вектором, состоящим из 8 комплексных чисел

#### 7.1.2 Функции

Функция для вычисления прямого ДП $\Phi$  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)
- 1		

# Возвращаемые значения

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

# Аргументы

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

7.2 FFT-16 21

# 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 БП $\Phi$  с плавающей точкой над вектором, состоящим из 16 комплексных чисел

#### Аргументы

#### Возвращаемые значения

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

#### Аргументы

$_{ m in}$	spec	структра, содержащая необходимые коэффициенты, для вычисления прямого БПФ
		определенного размера

#### 7.2.2.2 nmppsFFT16FwdInitAlloc\_32fcr()

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

# Аргументы

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

# Возвращает

7.3 FFT-32

### 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 БП $\Phi$  с плавающей точкой над вектором, состоящим из 32 комплексных чисел

#### Аргументы

in   х   входной вектор комплексных чисел (на мнимая и действительная части имеют тип
---

#### Возвращаемые значения

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

#### Аргументы

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

#### 7.3.2.2 nmppsFFT32FwdInitAlloc\_32fcr()

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

 $\Gamma$ руппы

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

# Аргументы

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

# Возвращает

7.4 FFT-64 25

### 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 БП $\Phi$  с плавающей точкой над вектором, состоящим из 64 комплексных чисел

#### Аргументы

in   х   входной вектор комплексных чисел (на мнимая и действительная части имеют тип
---

#### Возвращаемые значения

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

#### Аргументы

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

#### 7.4.2.2 nmppsFFT64FwdInitAlloc\_32fcr()

 $\Sigma$  Группы

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

# Аргументы

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

# Возвращает

7.5 FFT-128

### 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 )
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  с плавающей точкой над вектором, состоящим из 128 комплексных чисел

#### Аргументы

in   х   входной вектор комплексных чисел (на мнимая и действительная части имеют тип
---

#### Возвращаемые значения

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

#### Аргументы

$_{ m in}$	$\operatorname{spec}$	структра, содержащая необходимые коэффициенты, для вычисления прямого БПФ
		определенного размера

#### 7.5.2.2 nmppsFFT128FwdInitAlloc\_32fcr()

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

### Аргументы

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

# Возвращает

7.6 FFT-256

#### 7.6 FFT-256

#### Функции

• void nmppsFFT256Fwd\_32fcr (const nm32fcr \*x, nm32fcr \*X, NmppsFFTSpec\_32fcr \*spec) Функция для вычисления прямого БП $\Phi$  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 )
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  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	структра, содержащая необходимые коэффициенты, для вычисления прямого БП $\Phi$
		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()
```

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

### Аргументы

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

# Возвращает

7.7 FFT-512 31

#### 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()

```
\label{eq:const_nmpsffT512Fwd_32fcr} void \ nmppsFFT512Fwd_32fcr \ * \ x, \\ nm32fcr \ * \ X, \\ nm32fcr \ * \ X, \\ nmppsFFTSpec_32fcr \ * \ spec \ )
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  с плавающей точкой над вектором, состоящим из 512 комплексных чисел

#### Аргументы

	i	n	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	структра, содержащая необходимые коэффициенты, для вычисления прямого БП $\Phi$
		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()

```
\label{loc_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

7.8 FFT-1024 33

### 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()
```

```
\label{eq:condition} $\operatorname{void} \ nmppsFFT1024Fwd\_32fcr \ ($ const \ nm32fcr * x, $ nm32fcr * X, $ NmppsFFTSpec\_32fcr * spec \ )
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  с плавающей точкой над вектором, состоящим из 1024 комплексных чисел

#### Аргументы

in   х   входной вектор комплексных чисел (на мнимая и действительная части имеют тип
---

#### Возвращаемые значения

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

#### Аргументы

in	spec	структра, содержащая необходимые коэффициенты, для вычисления прямого Б $\Pi\Phi$
		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()
```

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

### Аргументы

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

# Возвращает

7.9 FFT-2048 35

### 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()

```
\label{eq:condition} $\operatorname{void} \ \operatorname{nmppsFFT2048Fwd} \_ 32 \mathrm{fcr} \ ($\operatorname{const} \ \operatorname{nm32fcr} * x,$$$ \operatorname{nm32fcr} * x,$$$ \operatorname{nm32fcr} * X,$$$ \operatorname{NmppsFFTSpec} \_ 32 \mathrm{fcr} * \mathrm{spec} \ )
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  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	структра, содержащая необходимые коэффициенты, для вычисления прямого Б $\Pi\Phi$
		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()

```
\label{loc_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

7.10 FFT-4096 37

#### 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()
```

```
\label{eq:const_nmpsfft} void \ nmppsFFT4096Fwd\_32fcr \ ( \ const\ nm32fcr * x, \ nm32fcr * X, \ NmppsFFTSpec \ 32fcr * spec \ )
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  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	структра, содержащая необходимые коэффициенты, для вычисления прямого БП $\Phi$
		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()
```

```
\label{loc_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

7.11 IDFT-8

# 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 (
```

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	структра, содержащая необходимые коэффициенты, для вычисления обратного БПФ
		определенного размера

40 Группы

## 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()
```

 $\Phi$ ункция для вычисления обратного БП $\Phi$  с плавающей точкой над вектором, состоящим из 16 комплексных чисел

#### Аргументы

m m modified berrop Rominionenian meeti (na minimali ii generalitettatian ideetii iimineet		in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
--	--	----	----	---

#### Возвращаемые значения

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

#### Аргументы

$_{ m in}$	ispec	структра, содержащая необходимые коэффициенты, для вычисления обратного БП $\Phi$
		определенного размера

#### 7.12.2.2 nmppsFFT16InvInitAlloc\_32fcr()

7.12 IFFT-16 41

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

## Аргументы

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

## Возвращает

## 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()
```

 $\Phi$ ункция для вычисления обратного БП $\Phi$  с плавающей точкой над вектором, состоящим из 32 комплексных чисел

#### Аргументы

m m modified berrop Rominionenian meeti (na minimali ii generalitettatian ideetii iimineet		in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
--	--	----	----	---

#### Возвращаемые значения

[out] іХ 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

#### 7.13.2.2 nmppsFFT32InvInitAlloc\_32fcr()

7.13 IFFT-32 43

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

## Аргументы

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

## Возвращает

44 Группы

#### 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()
```

 $\Phi$ ункция для вычисления обратного БП $\Phi$  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] іХ 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

#### 7.14.2.2 nmppsFFT64InvInitAlloc\_32fcr()

```
int nmppsFFT64InvInitAlloc_32fcr ( \frac{\text{NmppsFFTSpec}}{\text{NmppsFFTSpec}} = \frac{32\text{fcr}}{\text{standar}} * * \text{iaddr} )
```

7.14 IFFT-64 45

 $\Phi$ ункция инициализации структуры коэ $\phi$ фициентов, необходимых для вычисления обратного БП $\Phi$ -64.

## Аргументы

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

## Возвращает

## 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()
```

```
\label{eq:const_nm32fcr} void \ nmppsFFT128Inv\_32fcr \ ( const \ nm32fcr * ix, nm32fcr * iX, NmppsFFTSpec \ 32fcr * ispec \ )
```

 $\Phi$ ункция для вычисления обратного БП $\Phi$  с плавающей точкой над вектором, состоящим из 128 комплексных чисел

#### Аргументы

m m modified berrop Rominionenian meeti (na minimali ii generalitettatian ideetii iimineet		in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
--	--	----	----	---

#### Возвращаемые значения

[out] іХ 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

#### 7.15.2.2 nmppsFFT128InvInitAlloc\_32fcr()

7.15 IFFT-128 47

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

## Аргументы

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

## Возвращает

## 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()
```

```
\label{eq:const_nm32fcr} $$ void nmppsFFT256Inv\_32fcr ($$ const nm32fcr*ix, $$ nm32fcr*iX, $$ NmppsFFTSpec\_32fcr*ispec ) $$
```

 $\Phi$ ункция для вычисления обратного БП $\Phi$  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] іХ 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

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 ( \frac{\text{NmppsFFTSpec}}{\text{NmppsFFTSpec}} = \frac{32\text{fcr}}{\text{se}} ** \text{iaddr} )
```

7.16 IFFT-256 49

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

## Аргументы

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

## Возвращает

50 Группы

## 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()
```

 $\Phi$ ункция для вычисления обратного БП $\Phi$  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] іХ 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

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()
```

7.17 IFFT-512 51

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

## Аргументы

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

## Возвращает

## 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  ${\rm B}\Pi\Phi$  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] іХ 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

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 ( \frac{\text{NmppsFFTSpec}}{\text{NmppsFFTSpec}} = \frac{32\text{fcr}}{\text{se}} ** \text{iaddr} )
```

7.18 IFFT-1024 53

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

## Аргументы

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

## Возвращает

54 Группы

## 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 БП $\Phi$  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	t)
			-

#### Возвращаемые значения

[out] іХ 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

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()
```

7.19 IFFT-2048 55

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

## Аргументы

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

## Возвращает

56 Группы

## 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()
```

Функция для вычисления обратного БП $\Phi$  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] іХ 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

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

```
7.20.2.2 nmppsFFT4096InvInitAlloc 32fcr()
```

```
int nmppsFFT4096InvInitAlloc_32fcr ( \frac{\text{NmppsFFTSpec}}{\text{NmppsFFTSpec}} = \frac{32\text{fcr}}{\text{secondary}} ** \text{iaddr} )
```

7.20 IFFT-4096 57

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

## Аргументы

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

## Возвращает

58 Группы

## 7.21 FFT-Common

## Функции

```
• int nmppsFFTFwd 32fcr (const nm32fcr *x, nm32fcr *X, NmppsFFTSpec 32fcr *Spc)
```

 $\Phi$ ункция для вычисления прямого БП $\Phi$  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 БП $\Phi$  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()
```

```
 \begin{array}{c} int\ nmppsFFTFwd\_32fcr\ (\\ const\ nm32fcr\ *\ x,\\ nm32fcr\ *\ X,\\ NmppsFFTSpec\ 32fcr\ *\ Spc\ ) \end{array}
```

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	spec	структра, содержащая необходимые коэффициенты, для вычисления прямого БПФ
		определенного размера

#### 7.21.2.2 nmppsFFTFwdInitAlloc 32fcr()

7.21 FFT-Common 59 $\Phi$ ункция инициализации структуры коэффициентов, необходимых для вычисления прямого БП $\Phi$ с плавающей точкой над вектором длины от 8 до 2048.

## Аргументы

in	Spec	двойной указатель на структуру коэффициентов
in	Order	размерность БП $\Phi$ , которое нужно вычислить, например, для БП $\Phi$ 256 этот параметр равен 8 (т.к. $2^{\wedge}8=256$ )

## Возвращает

 $\Phi$ ункция возвращают 0 в случае успешной инициализации и отрицательное число (от -1 и меньше) в случае ошибок

7.22 IFFT-Common 61

## 7.22 IFFT-Common

## Функции

• int nmppsFFTInv 32fcr (const nm32fcr \*ix, nm32fcr \*iX, NmppsFFTSpec 32fcr \*iSpc)

 $\Phi$ ункция для вычисления обратного БП $\Phi$  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 БП $\Phi$  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*iSpc )
```

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()

62 $\Gamma$ руппы  $\Phi$ ункция инициализации структуры коэффициентов, необходимых для вычисления обратного БП $\Phi$ с плавающей точкой над вектором длины от 8 до 2048.

7.22 IFFT-Common 63

## Аргументы

in	iSpec	двойной указатель на структуру коэффициентов	
in		размерность БП $\Phi$ , которое нужно вычислить, например, для БП $\Phi$ 256 этот параметр равен 8 (т.к. $2^{\wedge}8=256$ )	

## Возвращает

 $\Phi$ ункция возвращают 0 в случае успешной инициализации и отрицательное число (от -1 и меньше) в случае ошибок

64 Группы

## 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 65

# 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  $nmpsMalloc_{-}$ .

# 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

 $7.25.2.1 \quad nm\_trans$ 

enum nm\_trans

Brief description.

Аргументы

in	n	Description for n
in	alpha	Description for alpha
in	arr⇔	Description for
	_x	arr_x
in	inc←	Description for
	_x	inc_x
in	arr⊷	Description for
	_у	arr_y
in	inc↔	Description for
	_у	inc_y

#### Возвращает

Return description

More details

См. определение в файле nmblas.h строка 61

7.25.3 Функции

7.25 BLASS-LEVEL1 67

```
7.25.3.1 nmblas_dasum()
```

```
\label{eq:const} \begin{array}{c} double \ nmblas\_dasum \ ( \\ & const \ int \ n, \\ & const \ double \ * \ x, \\ & const \ int \ inc\_x \ ) \end{array}
```

## Brief description.

## Аргументы

in	n	Description for n
in,out	x	Description for x
in	inc⊷	Description for
	_x	inc_x

# Возвращает

 ${\bf Return\ description}$ 

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 69

# 7.27 nmppcProdC

произведение двух комплексных чисел.

# Функции

## 7.27.1 Подробное описание

произведение двух комплексных чисел.

Аргументы

# 7.28 nmppcFixExp32

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

# Функции

• int nmppcFixExp32 (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 формату (16.16)

## Возвращает

Экспонента числа в формате с фиксрованной точкой (16.16)

# $7.29 \quad nmppcFixSinCos32$

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

#### Возвращаемые значения

pnSin	указатель на синус
pnCos	указатель на косинус

# 7.30 nmppcFixArcTan32

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

Функции

• int nmppcFixArcTan32 (int nArg)

## 7.30.1 Подробное описание

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

Аргументы

nArg | Угол в радианах

Возвращает

Арктангенс

# $7.31 \quad nmppcDoubleToFix 32$

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

arg	Входное число с плавающей точкой
fixpoint	позиция двоичной точки

#### Возвращает

Число с фиксрованной точкой

# 7.32 nmppcFix32ToDouble

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

## Функции

- double nmppcFix32ToDouble (int arg)

### 7.32.1 Подробное описание

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

Аргументы

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

#### Возвращает

Число с плавающей точкой

7.33 nmppcFixSqrt32

# 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 формату (16.16)

### Возвращает

Квадратный корень в формате с фиксрованной 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 формату (16.16)
nY	Второе входное число с фиксированной точкой в формату (16.16)

#### Возвращает

Произвдение в формате с фиксрованной точкой (16.16)

#### Аргументы

nX	Делимое
nY	Делитель

#### Возвращает

Частное от деления в формате с фиксрованной точкой (16.16)

7.35 nmppcFixInv 32

## 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 результирющем слове
--

### Возвращает

Частное от деления нав формате с фиксрованной точкой (16.16)

# 7.36 nmppcTblFixArcSin32

Вычисление функции arcsin по таблице. Входные и выходные значения задаются в формате fixedpoint (16.16)

## Функции

• int nmppcTblFixArcSin32 (int nArg)

## 7.36.1 Подробное описание

Вычисление функции arcsin по таблице. Входные и выходные значения задаются в формате fixedpoint (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 fixedpoint (16.16)

Аргументы

#### 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

 ${
m nArg}$   ${
m V}$ гол в диапазоне от -PI/2 до  ${
m +PI/2}$  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 nmppc FixDivPosMod32 (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 83

# 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

х Входное число с фиксированной точкой в формате (32.32)

### Возвращает

Квадратный корень в формате с фиксированной точкой (32.32)

# 7.42 nmppcDoubleToFix64

 $\Phi$ ункция 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 плвающей точкой типа double.

## Функции

• double nmppcFix64ToDouble (long arg, int fixpoint)

## 7.43.1 Подробное описание

Преобразование 64р. числа с фиксированной точкой в число с плвающей 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

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

## 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 nmppcFixArcTan64

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

Функции

• long nmppcFixArcTan64 (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 $^3$ 1; 2 $^3$ 1-1].

7.49 nmppcSqrt 91

# 7.49 nmppcSqrt

Вычисление квадратного корня

Функции

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

7.49.1 Подробное описание

Вычисление квадратного корня

Аргументы

х Входное число

Возвращает

Квадратный корень

# 7.50 Инициализация

# Группы

 $\bullet$  nmppsRand

Генерация случайного числа с равномерным распределением.

## 7.50.1 Подробное описание

# 7.51 Integer operations

# Группы

 $\bullet$  nmppcSqrt

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)

 $\bullet \ nmppcDoubleToFix 64$ 

Функция перевода из Fixed-Point 64 в Double.

 $\bullet$  nmppcFix64ToDouble

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

 $\bullet \ nmppcFixDiv 64$ 

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

• nmppcFixSinCos64

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

 $\bullet \ nmppcFixArcTan 64$ 

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

### 7.52.1 Подробное описание

7.53 Fix-point 32 95

## 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 плвающей 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 fixedpoint (16.16)

• nmppcTblFixSin32

Вычисление функции sin по таблице. Входные и выходные значения задаются в формате fixedpoint (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

 $\bullet$  nmppcDivC

частное двух комплексных чисел

• nmppcProdC

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()
```

Функция объединяет два полукдра в один полный кадр.

#### Аргументы

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

#### Возвращает

void

#### 7.55.2.2 IMG\_DeinterlaceSplit()

Функция разделяет кадр на два полукадра.

## Аргументы

pSrcImg	указатель на исходный кадр.
${ m nSrcWidth}$	ширина исходного кадра в пикселях.
${ m nSrcHeight}$	высота исходого 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

7.56.1 Подробное описание

Двумерная КИХ фильтрация

### 7.57 Floodfill

Исполняет разделение бинарной картинки на односвязные области. Пример вызова:  $no=VL\_ \leftarrow FloodFill32b(pSrcImage, Tetr,Image, pTmpBuff, nSrcWidth, nSrcHeight);$ 

### 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:  $no=VL\_ \leftarrow FloodFill32b(pSrcImage, Tetr,Image, pTmpBuff, nSrcWidth, nSrcHeight);$ 

#### Аргументы

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

#### Возвращаемые значения

pSegmentImage	Результирующее "изображение", где все точки одного сегмента имеют
	одинаковое значение (1,)

#### Возвращает

Число сегментов на изображении

#### Заметки

Maccub pSegmentImage дожен быть обнулен. Функция изменяет массив pSrcImage.

#### 7.57.2 Функции

7.57 Floodfill 101

#### 7.57.2.1 FloodFill8()

```
int FloodFill8 (
                  \mathrm{void} * \mathrm{src},
                  void * dst,
                  int nWidth,
                  int nHeight,
                  spot_struct * spot,
                  int lenSpot,
                  unsigned * pixels,
                  int mSpot,
                  int dtFull,
                  int dtSpot,
                  int lDiag,
                  int\ lDropSpot,
                  {\tt ds\_struct} * {\tt 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).

src	входная 8-битная матрица размером nHeight x nWidth. Внимание. Входная матрица src модифицируется:  • в целях оптимизации программы обнуляются верхняя и нижняя строки, и крайний левый и крайний правый столбцы (границы матрицы),  • в процессе обработки найденные пятна обнуляются.
dst	результирующая 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 солбца 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).

 $\Gamma$ руппы

lanCnat	20 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
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 пикселов, принадлежащих найденным пятнам, каждый элемент массива состоит из одного 32 битного слова, включающего в себя координаты пиксела в матрице src (или dst): unsigned int ij; биты [3116] - i - номер строки, биты [150] - 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 ріхеls, 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 солбца 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)," dtfull="" spot[0].="" t="" t0="" ymin="1." «нет»,="" в="" времени="" время,="" выход="" достаточность="" если="" задание="" задано="" замечание1.="" записью="" из="" на="" начало="" не="" обработку="" проверок="" программы="" программы).="" производится.="" происходит="" пятна="" работы="" с="" следующего="" текущее="" то="" –="">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?>
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).< td=""></dtfull).<>

7.57 Floodfill 103

lDiag	параметр, определяющий связность пятна по диагоналям:
	• lDiag=0 - элементы являются соседними, если они граничат или по вертикали, или по горизонтали (по ходам ладьи, но не по ходам слона),
	• lDiag=1 - элементы являются соседними, если они граничат или по вертикали, или по горизонтали, или по диагоналям (по ходам ферзя).
	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.
lDropSpot	параметр, задающий необходимость отбраковки найденных пятен по условиям НЕвхождения числа пикселов или размеров пятна в заданные диапазоны (см. ниже параметры nPxlmin, nPxlmax, dXYmin, dXYmax), статистика по отбракованным пятнам ведется в массиве dropSpot.
	IDropSpot=0 — найденные пятна НЕ отбраковываются, то есть все найденные пятна учитываются в массивах spot, pixels и отмечаются в выходной матрице dst, IDropSpot=1 — пятна с числом пикселов 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.  Замечание. Задание параметра IDropSpot=1 приводит к увеличению работы функции и требует выделения памяти для массива dropSpot. Если нет особой необходимости, задавайте IDropSpot=0.
dropSpot	– 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
	• 0 - dropSpot(nPxlMin): [ nnSpot, nnPxl, dttSpot]
	• 3 - dropSpot(nPxlMax): [ nnSpot, nnPxl, dttSpot]
	• 6 - dropSpot(dXYmin) : [ nnSpot, nnPxl, dttSpot]
	• 9 - dropSpot(dXYmax) : [ nnSpot, nnPxl, dttSpot]
	Перед обращением к функции массив dropSpot (размером 12 int) должен быть обнулен. При ldropSpot=0 статистика не ведется и массив dropSpot не требуется, то есть вместо него можно задать фиктивный параметр, например, 0.
nPxlMin	минимальное число пикселов в пятне, при lDropSpot=1: пятно отбраковывается, если число его пикселов меньше пPxlMin, при lDropSpot=0 проверок не производится, параметр можно задать 0.
nPxlMax	максимальное число пикселов в пятне,
	• при lDropSpot=1: пятно отбраковывается, если число его пикселов больше nPxlMax,
	• при lDropSpot=0 проверок не производится, параметр можно задать 0.

#### Аргументы

dXYmin	минимальная протяженность пятна по Х или по Ү в пикселах,
	• при lDropSpot=1: пятно отбраковывается, если его протяженность меньше dXYMax,
	• при lDropSpot=0 проверок не производится, но параметр используется для определения шага поиска по строкам входной матрицы src (и начальной строки поиска): dI = max(1, dXYmin)  Замечание. Задание dXYmin>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 парамера lDropSpot надо задавать dXYmin, по возможности, большим.
dXYmax	максимальная протяженность пятна по X или по Y в пикселах,  • при lDropSpot=1: пятно отбраковывается, если его протяженность больше dXYMax,  • при lDropSpot=0 проверок не производится, параметр можно задать 0.
	Ограничения: nPxlMin, nPxlMax, dXYmin, dXYmax - неотрицательные, (nPxlMin $<=$ nPxlMax), (dXYmin $<=$ dXYmax).

#### 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:

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.

7.57 Floodfill 105

#### 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» однопикселные пятна, расположенные по диагоналям максимальный размер массива spot.

#### Алгоритм

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

- пикселы каждого пятна занимают непрерывную область в массиве pixels,
- параметр spot[i].noPxl 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 noPxl = 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 пикселов пятна в массиве pixels: noPxl0 = noPxl1 = noPxl.
  - пиксел в матрице src обнуляется,
  - координаты пиксела заносятся в массив pixels, указатель noPxl1 увеличивается на 1 3. while-цикл обработки пятна:
  - выбирается пиксел noPxl0 из массива pixels, указатель noPxl0 увеличивается на 1
  - просматриваются все его соседние элементы (с учетом параметра lDiag) координаты ненулевых пикселов записываются в
  - пикселы в матрице src обнуляются
  - если указатели noPxl0, noPxl1 не равны (не все пикселы пятна и их соседи просмотрены), то переход на начало while-цик // пятно найдено! noPxl0=noPxl1 - указывают на начальный пиксел //следующего пятна в массиве pixels.

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

Eсли задан параметр IDropSpot=1, то проверка условий отбраковки пятна по признакам nPxlMin, nPxlMax, dXYmin, dXYmin, dXYmin если пятно отбраковано, то

формирование записи массива dropSpot по соответствующему признаку,

переход на поиск следующего пятна (noPxl не изменяется).

Меняется указатель noPxl=noPxl1.

 $\Phi$ ормирование параметров данного пятна в массиве spot[i] =

(Xmin, Ymin, Xmax, Ymax, noPxl, 0, 0, 0)

Если задан параметр вычисления угла наклона пятна (lAngleSpot=1), то вычисление codeAngle, lBadSpot\_и запись в spot[i] =

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

Выбор следующегл 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

pRGB	Вход, по 4 байта на пиксель. Порядок байтов В, G, R, 0. Результат в виде 32 битных
PICOB	- · · · · · · · · · · · · · · · · · · ·
	целых чисел, в которых полезные данные занимают младшие 24 бита. Для получения
	восьмибитовых пикселей неоходимо вырезать биты 1623, 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].

 $\Gamma$ руппы

# 7.60 Переупорядочивание изображений

# Группы

• Блочное переупорядочивание

Блочное перупорядочивание изображений.

7.60.1 Подробное описание

# 7.61 Блочное переупорядочивание

Блочное перупорядочивание изображений.

# Группы

 $\bullet \ IMG\_SplitIntoBlocks$ 

Преобразует изображение в последовательность квадратных блоков.

 $\bullet \ IMG\_MergeFromBlocks \\$ 

Объединяет последовательность квадратных блоков в изображение.

## 7.61.1 Подробное описание

Блочное перупорядочивание 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

pS	$\operatorname{rcImg}$	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

Выходная последовательность блоков имеет вид

```
 \begin{array}{l} [A00|A01|A02|...|A06|A07|A20|A21|...|A77|B00|B01|B02|...|B07|B10|...|H77|\\ [100|101|102|...|106|107]110|111|...|177|J00|J01|J02|........|Z77| \end{array}
```

# 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

```
 \begin{array}{l} [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| \end{array}
```

#### Результирующее изображение имеет вид

```
 \begin{bmatrix} \text{A00} & | \text{A01} & | \text{A02} & | \text{A03} & | \text{A04} & | \text{A05} & | \text{A06} & | \text{A07} & | \text{B00} & | \text{B01} \\ | \text{A11} & | \text{A11} & | \text{A12} & | \text{A13} & | \text{A14} & | \text{A15} & | \text{A16} & | \text{A17} & | \text{B10} & | \text{B11} \\ | \text{A20} & | \text{A21} & | \text{A22} & | \text{A23} & | \text{A24} & | \text{A25} & | \text{A26} & | \text{A27} & | \text{B20} & | \text{B21} \\ | \text{A30} & | \text{A31} & | \text{A32} & | \text{A33} & | \text{A34} & | \text{A35} & | \text{A36} & | \text{A37} & | \text{B30} & | \text{B31} \\ | \text{B37} & | \text{C30} & | \text{...} & | \text{H37} \\ | \text{...} & \text{...} & \text{...} & \text{...} \\ | \text{A70} & | \text{A71} & | \text{A72} & | \text{A73} & | \text{A74} & | \text{A75} & | \text{A76} & | \text{A77} & | \text{B70} & | \text{B71} \\ | \text{...} & \text{B77} & | \text{C70} & | \text{...} & | \text{H77} \\ | \text{...} & \text{...} & \text{...} & \text{...} \\ | \text{I00} & | \text{I01} & | \text{I02} & | \text{I03} & | \text{I04} & | \text{I05} & | \text{I06} & | \text{I07} & | \text{J00} & | \text{J01} \\ | \text{...} & \text{...} & \text{...} & \text{...} \\ | \text{I10} & | \text{I11} & | \text{I12} & | \text{I13} & | \text{I14} & | \text{I15} & | \text{I16} & | \text{I17} & | \text{J00} & | \text{I09} \\ | \text{...} & \text{...} & \text{...} & \text{...} \\ | \text{Z77} \end{bmatrix}
```

# 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  $\operatorname{IMG}$  \_Malloc.

 $7.65 \text{ IMG\_Release}$  113

# 7.65 IMG\_Release

Освобождение блоков памяти, выделенных функиями  ${\rm IMG\_Create}{***}.$ 

# Функции

# 7.65.1 Подробное описание

Освобождение блоков памяти, выделенных функиями  $IMG\_Create***$ .

114  $\Gamma$ руппы

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 функиями 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.

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

#### Аргументы

in	$\operatorname{str}$	String to be printed
in	img	Pointer to 8-bit grayscale image , where text should be printed
in	imgWidth	Image width
in	X	Horizontal position in image where text should be printed
in	у	Vertical position in image where text should be printed
in	FGcolor	Foreground text color
in	$\operatorname{BGcolor}$	Background text color

## Возвращает

x-coordinate behind printed text

#### Заметки

Russian text is supported in win-1251 codepage

 $\Gamma$ руппы

# 7.71 Инициализация и копирование

# Группы

• nmppmCopyua

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

•  $MTR\_Copyau$ 

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

• MTR Copy

Функции копирования прямоугольных областей памяти между двумерными массивами.

## 7.71.1 Подробное описание

7.72 nmppmCopyua 121

## 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 n ← DstStride, 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 n↔ Height, 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.
${ m nSrcOffset}$	Смещение в элементах от начала матрицы-источника.
nDstStride	Ширина результирующей матрицы в элементах.
nHeight	Число строк подматрицы.
nWidth	Число столбцов подматрицы.

#### Возвращаемые значения

#### 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 Copyau

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 n← DstOffset, 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

I	pSrcMtr	Исходная матрица.
1	${ m nSrcStride}$	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 123

## 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

#### 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

Возвращает

## 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 \*p  $\rightarrow$  DstVec, 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

# $7.77 \quad nmppmMul\_mv\_\_AddC$

Умножение матрицы на вектор с добавлением константы.

# Функции

• void nmppmMul\_mv\_\_AddC (v2nm32s \*pSrcMtr, v2nm32s \*pnSrcVec, int nAddVal, nm32s \*p  $\rightarrow$  DstVec, 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. $\mathrm{nHeight} = [0,2,4,].$
nWidth	Число столбцов исходной матрицы.

#### Возвращаемые значения

$\mathrm{pDstVec}$	Результирующий вектор.

#### Возвращает

# 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.
$_{ m nHeight}$	Число строк матрицы. nHeight=[128,256,]

#### Возвращаемые значения

pDstVec	Результирующий вектор.

Возвращает

7.79 MTR\_Malloc 129

# 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

# 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

## 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__ nm32u * MTR_Addr_32u (nm32u *pMTR, int nWidth, int nY, int nX)
__INLINE__ nm64u * MTR_Addr_32u (nm32u *pMTR, int nWidth, int nY, int nX)
__INLINE__ nm64u * MTR_Addr_32u (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

Зыписывате число в элемент марицы.

## 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, uint4b 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, uint16b nVal)
__INLINE__ void MTR_SetVal_32u (nm32u *pMtr, int nWidth, int nY, int nX, uint32b nVal)
__INLINE__ void MTR_SetVal_32u (nm32u *pMtr, int nWidth, int nY, int nX, uint32b 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

Зыписывате число в элемент марицы.

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

#### Аргументы

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

Возвращает

7.83 MTR GetVal 133

## 7.83 MTR GetVal

Считывает значение элемента марицы.

### 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, uint4b *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, uint16b *nVal)
__INLINE__ void MTR_GetVal_32u (nm32u *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_32u (nm32u *pMtr, int nWidth, int nY, int nX, uint4b *nVal)
```

#### 7.83.1 Подробное описание

Считывает значение элемента марицы.

nVal = pMtr[nY][nX]

#### Аргументы

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

Возвращаемые значения

nVal	Значение элемента	•
11 / 01	ona ichine onementa	

Возвращает

# 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.

 $\bullet \ \mathrm{MTR} \ \mathrm{SetVal}$ 

Зыписывате число в элемент марицы.

 $\bullet \ \mathrm{MTR\_GetVal}$ 

Считывает значение элемента марицы.

## 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.

 $\bullet \ \mathrm{MTR}\_\mathrm{ProdUnitV}$ 

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 \quad 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 преобразовние Фурье на базе алгоритма  $\mathbb{B}\Pi\Phi$  по основанию 16-16

Аргументы

in	$\operatorname{GSrcBuffer}$	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 6			
		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		
		установленна точность 7-бит функцией FFT Fwd256Set7bit() и 12 - если		
	ранее установлена точность 6-бит функцией FFT_Fwd256Set6bit(			

7.86 FFT-256

Возвращает

void

Заметки

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

${\it GSrcBuffer} \mid {\it LDstBuffer} \mid {\it LBuffer}  \mid {\it GBuffer}  \mid  {\it ShiftR}  \mid  {\it clocks}$								
L	L	L	L	-1	21.54			
L	L	L	G	-1	16.60			
L	L	G	L	-1	22.17			
L	L	Ğ	G	-1	20.98			
L	G	L	L	-1	20.53			
L	G	L	$\mathbf{G}$	-1	17.56			
L	G	$^{\mathrm{G}}$	L	-1	21.17			
L	G	$\mathbf{G}$	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	$\mathbf{G}$	0	16.58			
L	L	$_{\mathrm{G}}$	L	0	22.15			
L	L	G	$\mathbf{G}$	0	20.96			
L	$\mathbf{G}$	L	L	0	20.51			
L	G	L	$\mathbf{G}$	0	17.53			
L	G	G	L	0	21.14			
L	G	$\mathbf{G}$	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. ОБП $\Phi$ -256.

## 7.87.1 Подробное описание

## 7.87.2 Функции

Обратное быстрое преобразование Фурье. ОБПФ-256.

Функция выполняет обратное дискретное комплексное 256-точечное быстрое преобразование Фурье на базе алгоритма  $OB\Pi\Phi$  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). Неоходимо для предовтращения переполнения. По умолчанию равен 8
in	Shift R2	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 FFT_Inv256Set7bit() и 12 - при точности 6-бит, установленной с помощью функции FFT_Inv256Set6bit().

7.87 IFFT-256

## Возвращает

void

## Заметки

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

GSrcBufl	fer  GDst I	Buffer  LB	uffer   C	GBuffer	ShiftR1	ShiftR2	clocks
	L L L L L L L G G G G G L L L L L L L G G G G G G L L L L L L L G G G G G G L L L L L L L L G G G G G G L L L L L L L L G G G G G G L L L L L L L L G	L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L L L G G G G L L	LLGGLLGGLLGGLLGGLLGGLLGGLLGGLLGGLLGGLL	8 0 8 0 8 0 8 0 8 0 8 0 8 0 8 0 8 0 8 0	-1	26.76 26.76 19.07 19.07 22.87 22.87 22.87 22.87 20.77 20.77 25.75 25.75 18.06 18.06 23.82 23.82 21.72 21.72 25.80 25.80 18.10 18.10 21.91 21.91 19.80 19.80 24.79 24.79 17.09 17.09 17.09 22.86 22.86   20.76 26.74 26.74 19.04 19.04 22.85 22.85 20.75 20.75 25.73 18.03 18.03 23.80 23.80 21.70 21.70 25.77 18.08 18.08 21.88 21.88 19.78 19.78 19.78 24.76 24.76 17.07 17.07 22.83	

140  $\Gamma$ руппы

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

## 7.88 FFT-512

## Функции

```
• void FFT_Fwd512Set6bit ()
Устанавливает 6-битную точность вычислений
.
```

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 \quad 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 преобразовние Фурье на базе алгоритма  $\mathbb{B}\Pi\Phi$  по основанию 2-16-16

Аргументы

in	$\operatorname{GSrcBuffer}$	Входной массив размером 512 64-р. слов
out	${ m GDstBuffer}$	Результирующий массив размером 512 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 512*2 64-р. слов
in	in GBuffer Временный массив на глобальной шине (Global Bus) размером 512*3	
		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
		установленна точность 7-бит функцией FFT Fwd512Set7bit() и 12 - если
		ранее установлена точность 6-бит функцией FFT_Fwd512Set6bit().

 $\Gamma$ руппы

## Возвращает

void

## Заметки

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

GSrcBuffer   GDstBuffer   LBuffer   GBuffer   ShiftR   clock								
L	L	L	L	-1	24.12			
L	L	L	$_{\mathrm{G}}$	-1	19.29			
L	L	G	L	-1	22.81			
L	L	G	$\mathbf{G}$	-1	21.62			
L	$\mathbf{G}$	L	L	-1	23.10			
L	$\mathbf{G}$	L	$\mathbf{G}$	-1	18.27			
L	$\mathbf{G}$	G	L	-1	23.77			
L	$\mathbf{G}$	$_{ m G}$	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	$\mathbf{G}$	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			
$\mathbf{G}$	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	$\mathbf{G}$	0	21.61			
L	$\mathbf{G}$	L	L	0	23.09			
L	$\mathbf{G}$	L	$\mathbf{G}$	0	18.26			
L	$\mathbf{G}$	G	L	0	23.76			
L	$\mathbf{G}$	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	$\mathbf{G}$	0	23.55			

7.89 IFFT-512

## 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 \quad 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.

 $\Phi$ ункция выполняет обратное дискретное комплексное 512-точечное быстрое преобразование  $\Phi$ урье на базе алгоритма ОБП $\Phi$  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). Неоходимо для предовтращения 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 FFT_Inv512Set7bit() и 12 - при точности 6-бит, установленной с помощью функции FFT_Inv512Set6bit().

 $\Gamma$ руппы

## Возвращает

void

Заметки

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

GSrcBuff	fer  LDst E	Buffer  LE	Buffer   C	GBuffer	ShiftR1	ShiftR2	clocks
L	L	L	L	9	- 1	23.41	
L	L	L	L	0	-1 -1	$\frac{23.41}{23.40}$	
L	L	L	G G		-1   -1		
				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	
$_{\widetilde{\alpha}}$	L	L	L	9	-1	22.35	
$_{\widetilde{\alpha}}$	L	L	L	0	-1	22.34	
G	L	L	G	9	-1	18.34	
$_{\widetilde{\alpha}}$	L	L	G	0	-1	18.33	
$_{\widetilde{G}}$	L	G	L	9	-1	25.84	
$_{\widetilde{\alpha}}$	L	G	L	0	-1	25.84	
G	L	G	G	9	-1	27.45	
$\mathbf{G}$	L	G	G	0	-1	27.44	
$\mathbf{G}$	G	L	L	9	-1	21.33	
$\mathbf{G}$	G	L	L	0	-1	21.33	
$\mathbf{G}$	G	L	G	9	-1	19.30	
$\mathbf{G}$	G	L	G	0	-1	19.29	
$\mathbf{G}$	G	G	L	9	-1	24.83	
$\mathbf{G}$	G	G	L	0	-1	24.82	
$\mathbf{G}$	G	G	$\mathbf{G}$	9	-1	28.41	
$\mathbf{G}$	G	G	$\mathbf{G}$	0	-1	28.41	
L	L	L	L	9	0	23.40	
L	L	L	L	0	0	23.39	
L	L	L	$\mathbf{G}$	9	0	19.38	
L	L	L	G	0	0	19.37	
L	L	$\mathbf{G}$	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	
$\mathbf{G}$	L	L	L	9	0	22.34	
$\mathbf{G}$	L	L	L	0	0	22.33	
$\mathbf{G}$	L	L	$^{\mathrm{G}}$	9	0	18.33	
$\mathbf{G}$	L	L	G	0	0	18.32	
$\mathbf{G}$	L	G	L	9	0	25.83	
G	L	G	L	0	0	25.83	
$\mathbf{G}$	L	G	G	9	0	27.43	
$\mathbf{G}$	L	G	$\mathbf{G}$	0	0	27.43	
$\mathbf{G}$	G	L	L	9	0	21.32	
$\mathbf{G}$	G	L	L	0	0	21.31	
$\mathbf{G}$	G	L	G	9	0	19.29	
G	G	L	G	0	0	19.28	
G	G	G	L	9	0	24.81	

7.89 IFFT-512

G	G	G	L	0	0	24.81
G	G	G	G	9	0	28.40
G	$\mathbf{G}$	$\mathbf{G}$	G	i 0	i o	28 39

 $\Gamma$ руппы

## 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 \quad 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 преобразовние Фурье на базе алгоритма  $\mathbb{B}\Pi\Phi$  по основанию 2--32--16

Аргументы

in	$\operatorname{GSrcBuffer}$	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 установленна точность 7-бит функцией FFT_Fwd1024Set7bit() и 12 - если ранее установлена точность 6-бит функцией FFT_Fwd1024Set6bit().

7.90 FFT-1024

Возвращает

void

Заметки

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

GSrcBuff	fer  LDstB	suffer  LB	uffer   G	Buffer	ShiftR	clocks
L	L	L	L	-1	22.55	
L	Ĺ	L	G	-1	20.52	
L	L	G	L	-1	26.08	
L	L	G	$^{\mathrm{G}}$	-1	25.93	
L	G	L	L	-1	21.53	
L	$\mathbf{G}$	L	G	-1	21.48	
L	$\mathbf{G}$	G	L	-1	25.06	
L	$\mathbf{G}$	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	$^{\mathrm{G}}$	-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	$\mathbf{G}$	L	$\mathbf{G}$	0	21.48	
L	$\mathbf{G}$	G	L	0	25.05	
L	$\mathbf{G}$	G	$\mathbf{G}$	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	
$\mathbf{G}$	G	G	L	0	26.00	
$\mathbf{G}$	G	$\mathbf{G}$	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 \quad 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  $OB\Pi\Phi$  по онованию 2-16-16.

### Аргументы

in	$\operatorname{GSrcBuffer}$	Входной массив размером 1024 64-р. слов
out	${ m GDstBuffer}$	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). Неоходимо для предовтращения 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 FFT_Inv1024Set7bit() и 12 - при точности 6-бит, установленной с помощью функции FFT_Inv1024Set6bit().

7.91 IFFT-1024 149

## Возвращает

void

### Заметки

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

CSnoBufford CDatl	Duffor I Duffor	CDuffer   C	%;#D1   Ck;#D0	ala alsa
CSrcBuffer   GDst   GDst	L	10   0   10   10   10   10   10   10	Shift R1   Shift R2	clocks
$G \mid G$	G   L	10	0   28.51	

150 Группы

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

7.92 FFT-2048

## 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 \quad 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 преобразовние Фурье на базе алгоритма БП $\Phi$  по основанию 2-32-32

### Аргументы

in	GSrcBuffer	Входной массив размером 2048 64-р. слов
out	${ m GDstBuffer}$	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 LDstBuffer. При передаче значения по умолчанию (-1) ShiftR автоматически принимается равным 14 если ранее установленна точность 7-бит функцией FFT_Fwd2048Set7bit() и 12 - если ранее установлена точность 6-бит функцией FFT_Fwd2048Set6bit().

Возвращает

void

152 Группы

Заметки

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

$ \label{eq:GSrcBuffer}                                   $					
L	L	L	-1	26.41	
L	L	G	-1	28.97	
L	$\mathbf{G}$	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	$_{\mathrm{G}}$	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

## 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

Обратное быстрое преобразование Фурье. ОБПФ-2048.

Функция выполняет обратное дискретное комплексное 2048-точечное быстрое преобразование Фурье на базе алгоритма  $OB\Pi\Phi$  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). Неоходимо для предовтращения 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 FFT_Inv2048Set7bit() и 12 - при точности 6-бит, установленной с помощью функции FFT_Inv2048Set6bit().

Создано системой Doxygen

 $\Gamma$ руппы

## Возвращает

void

### Заметки

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

GSrcBuff	fer  LDstF	Buffer  LE	Buffer	GBuffer	Shift R1	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	$\mathbf{G}$	0	-1	26.52	
L	L	$^{\mathrm{G}}$	L	11	- 1	31.33	
L	$_{ m L}$	$\mathbf{G}$	L	0	-1	31.33	
L	L	G	$\mathbf{G}$	11	-1	29.26	
L	L	$\mathbf{G}$	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	
$_{ m L}$	G G	L	G	0	-1	27.48	
L	G	G G	$egin{array}{ccc} & & \mathrm{L} & & \\ & & \mathrm{L} & & \end{array}$	11	-1 -1	30.30	
L	G	G	G	1 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	
$\mathbf{G}$	L	L	G	11	-1	25.49	
$\mathbf{G}$	L	L	G	0	-1	25.49	
$\mathbf{G}$	L	G	L	11	-1	32.29	
$\mathbf{G}$	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	
$_{\widetilde{G}}$	G	L	L	0	-1	28.53	
G	G	L	G	11	-1	26.46	
G	G	L	G	0	-1	26.46	
$_{ m G}$	G G	G   G	$egin{array}{ccc} oldsymbol{\mathrm{L}} & oldsymbol{\mathrm{L} & oldsymbol{\mathrm{L}} & oldsymbol{\mathrm{L} & oldsymbol{\mathrm{L}} & oldsymbol{\mathrm{L}} & oldsymbol{\mathrm{L}} & oldsymbol{\mathrm{L} & oldsymbol{\mathrm{L}$	11	-1	31.26	
G	G	G	G	0   11	-1	$\begin{vmatrix} 31.26 \\ 31.19 \end{vmatrix}$	
G	G	G	G	0	-1	31.18	
L	L	L I	L	1 11	0	30.58	
Ĺ	L	L	Ĺ	0	0	30.58	
L	L	L	G	11	0	26.51	
L	L	L	$\mathbf{G}$	0	0	26.51	
L	L	$^{\mathrm{G}}$	L	11	0	31.33	
L	L	$\mathbf{G}$	L	0	0	31.32	
L	L	G	$\mathbf{G}$	11	0	29.25	
L	L	$\mathbf{G}$	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	
$_{ m L}$	G G	G G	$egin{array}{ccc} & & \mathrm{L} & & \\ & & \mathrm{L} & & \end{array}$	11 0	0 0	30.30 30.30	
L	G	G	G	1 11	0	30.22	
L	G	G	G	0		30.22	
G	L	L	L	11	0	29.56	
$\tilde{ ext{G}}$	Ĺ	L	Ĺ	0	0	29.56	
$\mathbf{G}$	L	L	G	11	0	25.49	
$\mathbf{G}$	L	L	G	0	0	25.49	
$\mathbf{G}$	L	G	L	11	0	32.29	
$\mathbf{G}$	L	G	L	0	0	32.29	
$\mathbf{G}$	L	G	G	11	0	30.22	
$\mathbf{G}$	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	

7.93 IFFT-2048 155

G	G	G	L	0	0	31.26
G	G	G	G	11	0	31.18
G	$\mathbf{G}$	G	$\mathbf{G}$	0	i 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 \quad FFT\_Fwd4096() void \ FFT\_Fwd4096 \ ( nm32sc * GSrcBuffer, nm32sc * GDstBuffer, void * LBuffer, void * GBuffer )
```

Прямое быстрое преобразование Фурье-4096.

Функция выполняет дискретное комплексное 4096-точечное преобразовние Фурье на базе алгоритма БП $\Phi$  по основанию 16-16-16

### Аргументы

in	$\operatorname{GSrcBuffer}$	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

 $\backslash perf$ 

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

L	L	L	L	38.25
L	L	L	$_{\mathrm{G}}$	26.82
L	L	G	L	32.21
L	L	G	G	30.74

7.94 FFT-4096

L	G	L	$_{\rm L}$	37.22
L	G	L	$\mathbf{G}$	25.79
L	G	G	L	33.17
L	G	$^{\mathrm{G}}$	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	$\mathbf{G}$	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

Обратное быстрое преобразование Фурье. ОБПФ-4096.

Функция выполняет обратное дискретное комплексное 4096-точечное быстрое преобразование Фурье на базе алгоритма  $OB\Pi\Phi$  по онованию 16-16-16.

### Аргументы

in	GSrcBuffer	Входной массив размером 4096 64-р. слов
out	${ m GDstBuffer}$	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

 $\backslash perf$ 

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

L	l L	l L l	L	38.25
L	L	L	$\mathbf{G}$	26.82
L	L	G	L	32.21
L	L	G	G	30.74

7.95 IFFT-4096 159

L	G	L	$_{\rm L}$	37.22
L	G	L	$\mathbf{G}$	25.79
L	G	G	L	33.17
L	G	$\mathbf{G}$	$\mathbf{G}$	31.71
G	L	L	L	37.26
G	L	L	$\mathbf{G}$	25.83
G	L	G	L	31.21
G	L	G	$\mathbf{G}$	29.75
G	G	L	L	36.23
G	G	L	G	24.80
G	G	G	L	32.18
G	$\mathbf{G}$	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

Прямое быстрое преобразование Фурье-8192.

Функция выполняет дискретное комплексное 8192-точечное преобразовние Фурье на базе алгоритма БП $\Phi$  по основанию 2-16-16

### Аргументы

in	LSrcBuffer	Входной массив размером 8192 64-р. слов
out	${ m GDstBuffer}$	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

 $\backslash perf$ 

LSrcBuffe	er   GDst1	Buffer   L	Buffer	GBuffer   Clocks
L	L	L	L	40.70
L	L	L	$\mathbf{G}$	28.89
L	L	G	L	35.55
L	L	G	G	31.88
T.	G	Τ.	Т.	1 39 67

7.96 FFT-8192

L	G	L	$\mathbf{G}$	27.86
L	G	G	L	36.52
L	G	$\mathbf{G}$	G	32.85
G	L	L	L	40.17
G	L	L	G	29.40
G	L	G	L	35.02
G	L	G	$^{\mathrm{G}}$	32.39
G	G	L	L	39.14
G	G	L	$^{\mathrm{G}}$	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

Обратное быстрое преобразование Фурье. ОБПФ-8192.

Функция выполняет обратное дискретное комплексное 8192-точечное быстрое преобразование Фурье на базе алгоритма  $OB\Pi\Phi$  по онованию 2-16-16.

### Аргументы

in	LSrcBuffer	Входной массив размером 8192 64-р. слов
out	${ m GDstBuffer}$	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

 $\backslash p\,erf$ 

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

L	L	L	L	40.70
L	L	L	G	28.89

7.97 IFFT-8192

L	L	G	L	35.55
L	L	G	$^{\mathrm{G}}$	31.88
L	$\mathbf{G}$	L	L	39.67
L	$\mathbf{G}$	L	$\mathbf{G}$	27.86
L	$\mathbf{G}$	G	L	36.52
L	$\mathbf{G}$	G	G	32.85
G	L	L	L	40.17
G	L	L	$\mathbf{G}$	29.40
G	L	$^{\mathrm{G}}$	L	35.02
G	L	$^{\mathrm{G}}$	G	32.39
G	G	L	L	39.14
G	G	L	G	28.37
G	G	G	L	35.99
G	G	G	$\mathbf{G}$	33.36

 $\Gamma$ руппы

# 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.

### $\bullet \ SIG\_Resample\_perf$

Функции для оценки производительности функций фильтрации SIG \_Resample()

Функция эквивалентно следующим псевдоинструкциям:

### 7.100.1 Подробное описание

## 7.101 Быстрое преобразование Фурье

## Группы

• DFT-8

Функция для вычисления прямого ДП $\Phi$  с плавающей точкой над вектором, состоящим из 8 комплексных чисел

- FFT-16
- FFT-32
- FFT-64
- FFT-128
- FFT-256
- FFT-512
- 111 012
- FFT-1024FFT-2048
- FFT-4096
- IDFT-8
- IFFT-16
- IFFT-32
- IFFT-64
- IFFT-128
- IFFT-256
- IFFT-512
- IFFT-1024
- IFFT-2048
- IFFT-4096
- FFT-Common
- IFFT-Common

### Функции

• int nmppsFFTFree 32fcr (NmppsFFTSpec 32fcr \*spec)

 $\Phi$ ункция освобождает память, выделенную под коэффициенты, необходимые для вычисления БПФ определенного размера

### 7.101.1 Подробное описание

### 7.101.2 Функции

```
7.101.2.1 nmppsFFTFree_32fcr() int nmppsFFTFree_32fcr (
```

NmppsFFTSpec\_32fcr \* spec )

 $\Phi$ ункция освобождает память, выделенную под коэффициенты, необходимые для вычисления БПФ определенного размера

 $\Gamma$ руппы

## Аргументы

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

## Возвращает

 $\Phi$ ункция возвращают 0 в случае успешной инициализации и отрицательное число (от -1 и меньше) в случае ошибок

## 7.102 Элементарные математические функции

### Группы

• nmppsSin 32f

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

• nmppsCos 32f

Функция, вычисляющая косинус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

• nmppsDiv 32f

Функция, вычисляющая x/y (pSrcVec1 / pSrcVec2), где x и y (делимые числа и делители) это вектора чисел с плавающей точкой одинарной точности

• nmppsExp 32f

Функция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

• nmppsExp 64f

Функция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

• nmppsLog 32f

Функция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

• nmppsLog 64f

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

• nmppsPowx 64f

Функция возводит в степень (Deg) каждый элемент вектора чисел с плавающей точкой двойной точности (64f)

### 7.102.1 Подробное описание

## 7.103 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)

### 7.103.1 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 буффер размера 2*nKernelSize + 32 32-битных слов;	

### 2\*nKernelSize + 32 32-bit words;

### Аргументы

nKernelSize	Размер окна свертки [1,2,3,4nSrcVecSize-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	
	записыватсья еще до 7 незначачих 32р слова.	

 $7.103 \ \mathrm{SIG\_XCorr}$  171

По сути функции осуществляют фильтрацию данных окном свертки.

## 7.104 nmppsSin 32f

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### Функции

• void nmppsSin\_32f (const nm32f \*pSrcVec, nm32f \*pDstVec, int nSize)

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинар-

7.104.1 Подробное описание

ной точности (32f)

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

7.104.2 Функции

```
7.104.2.1 \quad nmppsSin\_32f()

void nmppsSin_32f (

const nm32f * pSrcVec,

nm32f * pDstVec,

int nSize )
```

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

Аргументы

in	pSrcVec	входной вектор чисел с плавающей точкой

Возвращаемые значения

[out] рDstVec выходной вектор чисел с плавающей точкой

Аргументы

in nSize число элементов в векторе

7.105 nmppsCos 32f

## 7.105 nmppsCos 32f

 $\Phi$ ункция, вычисляющая косинус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### Функции

• void nmppsCos\_32f (const nm32f \*pSrcVec, nm32f \*pDstVec, int nSize)

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### 7.105.1 Подробное описание

 $\Phi$ ункция, вычисляющая косинус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### 7.105.2 Функции

```
7.105.2.1 nmppsCos_32f()

void nmppsCos_32f (
	const nm32f * pSrcVec,
	nm32f * pDstVec,
	int nSize )
```

Функция, вычисляющая синус над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

Аргументы

$_{ m in}$	pSrcVec	входной вектор чисел с плавающей точкой одинарной точности

#### Возвращаемые значения

[out] рDstVec выходной вектор чисел с плавающей точкой одинарной точности

#### Аргументы

in	nSize	число элементов в векторе	

## 7.106 nmppsDiv 32f

 $\Phi$ ункция, вычисляющая х/у (pSrcVec1 / pSrcVec2), где х и у (делимые числа и делители) это вектора чисел с плавающей точкой одинарной точности

### Функции

• void nmppsDiv\_32f (const nm32f \*pSrcVec1, const nm32f \*pSrcVec2, nm32f \*pDstVec, int nSize) Функция, вычисляющая х/у (pSrcVec1 / pSrcVec2), где х и у (делимые числа и делители) это вектора чисел с плавающей точкой одинарной точности

### 7.106.1 Подробное описание

 $\Phi$ ункция, вычисляющая x/у (pSrcVec1 / pSrcVec2), где x и у (делимые числа и делители) это вектора чисел с плавающей точкой одинарной точности

### 7.106.2 Функции

```
7.106.2.1 \quad nmppsDiv\_32f() void \; nmppsDiv\_32f() const \; nm32f * pSrcVec1, const \; nm32f * pSrcVec2, nm32f * pDstVec, int \; nSize)
```

 $\Phi$ ункция, вычисляющая x/у (pSrcVec1 / pSrcVec2), где x и у (делимые числа и делители) это вектора чисел с плавающей точкой одинарной точности

### Аргументы

iı	n	pSrcVec1	входной вектор делимых чисел с плавающей точкой одинарной точности
iı	n	${ m pSrcVec2}$	входной вектор делителей с плавающей точкой одинарной точности

#### Возвращаемые значения

[out] рDstVec выходной вектор частных с плавающей точкой одинарной точности
---

### Аргументы

in	nSize	число элементов в векторе

7.107 nmppsExp 32f 175

## 7.107 nmppsExp 32f

 $\Phi$ ункция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### Функции

• void nmppsExp 32f (const nm32f \*pSrcVec, nm32f \*pDstVec, int nSize)

 $\Phi$ ункция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### 7.107.1 Подробное описание

 $\Phi$ ункция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### 7.107.2 Функции

 $\Phi$ ункция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

Аргументы

$_{ m in}$	pSrcVec	входной вектор чисел с плавающей точкой одинарной точности

#### Возвращаемые значения

[out] рDstVec выходной вектор чисел с плавающей точкой одинарной точности

### Аргументы

in	nSize	число элементов в векторе	

## 7.108 nmppsExp 64f

 $\Phi$ ункция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### Функции

• void nmppsExp 64f (const nm64f \*pSrcVec, nm64f \*pDstVec, int nSize)

Функция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### 7.108.1 Подробное описание

Функция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### 7.108.2 Функции

 $\Phi$ ункция, вычисляющая экспоненту над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

Аргументы

in	pSrcVec	входной вектор чисел с плавающей точкой двойной точности

#### Возвращаемые значения

[out] рDstVec выходной вектор чисел с плавающей точкой двойной точности

#### Аргументы

in	nSize	число элементов в векторе	

7.109 nmppsLog 32f 177

## 7.109 nmppsLog 32f

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### Функции

• void nmppsLog 64f (const nm32f \*pSrcVec, nm32f \*pDstVec, int nSize)

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### 7.109.1 Подробное описание

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

### 7.109.2 Функции

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой одинарной точности (32f)

Аргументы

in	pSrcVec	входной вектор чисел с плавающей точкой одинарной точности

Возвращаемые значения

[out] рDstVec выходной вектор чисел с плавающей точкой одинарной точности

### Аргументы

in	nSize	число элементов в векторе	

## 7.110 nmppsLog 64f

Функция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### Функции

• void nmppsLog 64f (const nm64f \*pSrcVec, nm64f \*pDstVec, int nSize)

Функция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### 7.110.1 Подробное описание

Функция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### 7.110.2 Функции

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

Аргументы

in	pSrcVec	входной вектор чисел с плавающей точкой двойной точности

#### Возвращаемые значения

[out] рDstVec выходной вектор чисел с плавающей точкой двойной точности

#### Аргументы

in	nSize	число элементов в векторе	e

## 7.111 nmppsPowx 64f

Функция возводит в степень (Deg) каждый элемент вектора чисел с плавающей точкой двойной точности (64f)

### Функции

• void nmppsPowx\_64f (const nm64f \*pSrcVec, nm64f \*pDstVec, nm32u Deg, int nSize) Функция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### 7.111.1 Подробное описание

 $\Phi$ ункция возводит в степень (Deg) каждый элемент вектора чисел с плавающей точкой двойной точности (64f)

### 7.111.2 Функции

 $\Phi$ ункция, вычисляющая логарифм над каждым элементом вектора чисел с плавающей точкой двойной точности (64f)

### Аргументы

in	pSrcVec	входной вектор чисел с плавающей точкой двойной точности
----	---------	--

#### Возвращаемые значения

[out]	pDstVec выходной вектор чисел с плавающей точкой двойной точности
-------	---

### Аргументы

in	Deg	степень, в которую возводится каждый элемент входного вектора
in	nSize	число элементов в векторе

180 Группы

#### SIG\_Median3 7.112

Вычисление медианы трех чисел

## Функции

- int SIG\_Median3 (int a, int b, int c)
  uint32b SIG\_Median3 (uint32b a, uint32b b, uint32b c)

## 7.112.1 Подробное описание

### Вычисление медианы трех чисел

### Аргументы

a	Первое число
b	Второе число
С	Третье число

### Возвращает

Медианное значение

# 7.113 КИХ-фильтрация

Одномерная КИХ-фильтрация.

### Группы

• nmppsFIR\_Xs

Одномерная КИХ-фильтрация

 $\bullet \ nmppsFIRInit\_Xs$ 

Инициализация функции одномерной фильтрации

 $\bullet \ nmppsFIRInitAlloc\_Xs$ 

Выделение и инициализация служебной структуры для функции одномерной фильтрации

 $\bullet \ nmppsFIRGetStateSize\_Xs$ 

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

• nmppsFIRFree

освобождает структуру pState в куче

### 7.113.1 Подробное описание

Одномерная КИХ-фильтрация.

# 7.114 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)

#### 7.114.1 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  $\leftarrow$  Init\_Xs() или nmppsFIRInitAlloc\_Xs(). Максимальная производительность достигается при размещении pSrc, pDst и pPstate в разных банках памяти .

## 7.115 nmppsFIRInit Xs

Инициализация функции одномерной фильтрации

### Функции

- int nmpps FIRInit\_8s (Nmpps FIRState \*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)

### 7.115.1 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 32рсловах) необходимый для хранения служебной структуры можно получить с помощью функции nmppsFIRGetStateSize_Xs

#### Возвращает

Размер проинициализированной структуры pState в 32p. словах

# 7.116 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)

#### 7.116.1 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 32p. словах

# 7.117 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)

#### 7.117.1 Подробное описание

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

#### Аргументы

#### Возвращает

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

# 7.118 nmppsFIRFree

освобождает структуру pState в куче

# Функции

 $\bullet \ \ void \ nmppsFIRFree \ (NmppsFIRState *pState)$ 

### 7.118.1 Подробное описание

освобождает структуру pState в куче

Аргументы

in pState указатель на служебную структуру NmppsFIRState

# 7.119 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)

### 7.119.1 Подробное описание

Уменьшение числа отсчетов в двое.

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

#### Аргументы

pSrcVec	Входной сигнал.
nSize	Размер массива входных данных.

### Возвращаемые значения

$\mathrm{pDstVec}$	Результирующий сигнал.

### Заметки

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

# $7.120 \quad SIG\_ResampleUp3Down2$

Передискретизации сигнала в 3/2 раза

Передискретизации сигнала осуществляется методом Polyphase:

.

### Функции

• void SIG\_Resample Up3Down2 (nm8s \*pSrcVec, nm16s \*pDstVec, int nSrcVecSize, nm64s \*p  $\leftarrow$  Kernel)

### 7.120.1 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

 $\operatorname{void}$ 

# 7.121 SIG CreateResample

Создание ядра для функции передискретизации SIG Resample().

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

### Функции

- $\bullet \ \ void \ SIG\_CreateResampleUp3Down2\_8s16s \ (nm64s \ **pKernel, int \ nHint=MEM\_LOCAL)$
- void SIG CreateResampleDown2 8u8u (nm64s \*\*pKernel, int nHint=MEM LOCAL)
- void SIG CreateResampleDown2 16u16u (nm64s \*\*pKernel, int nHint=MEM LOCAL)

### 7.121.1 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 коэффиценты фильтра во
	внутреннем формате.

#### Заметки

Используется перед вызовом функции SIG Filter.

# 7.122 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)

### 7.122.1 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 коэффиценты фильтра во
	внутреннем формате.

#### Возвращает

Размер проинициализированной структуры pKernel в 32p. словах

#### Заметки

Используется перед вызовом функции SIG\_Filter. Используется перед вызовом функции SIG Resample().

## 7.123 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)

#### 7.123.1 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);
}
```

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

#### Аргументы

 $\retval$ 

pSrcVec	Входной вектор.
${\rm nSrcVecSize}$	Размер входного вектора в элементах.

#### Возвращаемые значения

pKernel

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

### Заметки

Инициализация служебной структуры производится соответсвующей функцией  $SIG\_Set \leftarrow Filter()$  и  $SIG\_CreateFilter()$ .

## 7.124 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
- $\bullet \ \, {\rm typedef} \,\, {\rm void} \,\, {\rm nm2u}$
- $\bullet$  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 v16nm8sv16nm8s7b
- 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
- · typeder struct s\_vrommrod vrommro
- typedef struct  $s_v2nm32u$  v2nm32u
- typedef struct s\_v4nm32u v4nm32u
- typedef struct s v8nm32u v8nm32u
- typedef struct s v16nm32u v16nm32u
- typedef v16nm4u v16nm4b3u

#### 7.124.1 Подробное описание

В данном разделе описываются типы векторных данных с которыми могут работать функции библиотеки, задействующие векторный узел. Также рассматриваюся 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 являтся 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 обозначания 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 выполянться некорректно.

Расшифровка мнемоники в названии типа:

1. Префикс nm - означет что данные являются векторными ,упакованными в 64р слова (nm8s,nm8u,nm16s....).

- 2. Разрядность данных указывается после префикса nm (nm8s,nm8u байтовые массивы, nm16s,nm16u 16p. массивы ).
- 3. суфикс в или и оначает знаковый или беззнаковый тип данных.
- 4. Для работы некоторых функций во избежании переполнения требуется суженный диапазон возможных значений, чем позволяет разрядность. Такие даные имеют суффикс в виде кол-ва значащих бит в слове и символом b. (nm8s7b)

7.124.2 Типы

7.124.2.1 nm1

typedef void nm1

Большинство функций библиотеки получают и возвращают массивы упакованных чисел. Обращение к элементам данных массивов должно производится 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

7.124.2.2 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:

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

 $7.124.2.3 \quad nm16s15b$ 

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:

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

См. определение в файле nmtype.h строка 231

7.124.2.4 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:

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

См. определение в файле nmtype.h строка 444

7.124.2.5 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:

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

 $7.124.2.6 \, \text{nm2s}$ 

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 nmtype.h строка 113

7.124.2.7 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:

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

См. определение в файле nmtype.h строка 369

7.124.2.8 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:

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

 $7.124.2.9 \quad nm32s30b$ 

typedef int nm32s30b

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:

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

См. определение в файле nmtype.h строка 319

7.124.2.10 nm32s31b

typedef int nm32s31b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 299

7.124.2.11 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 64р слова. Предполагается, что размер массива данного типа как минимум кратен 2.

Диапазон значений

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

7.124.2.12 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 64р слова. Предполагается, что размер массива данного типа как минимум кратен 2.

Диапазон значений

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

См. определение в файле nmtype.h строка 488

7.124.2.13 nm4s

typedef void nm4s

Тип характеризует векторные данные как массив 4-х разрядных чисел со знаком. Начальный адрес массива должен быть выровнен по границе 64р 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 nmtype.h строка 126

7.124.2.14 nm4u

typedef void nm4u

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

Диапазон значений:

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

 $7.124.2.15 \quad nm4u3b$ 

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:

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

См. определение в файле nmtype.h строка 395

7.124.2.16 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:

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

См. определение в файле nmtype.h строка 340

7.124.2.17 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:

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

7.124.2.18 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 64р слова. По умолчанию размер массива произвольный .

Диапазон значений

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

См. определение в файле nmtype.h строка 501

7.124.2.19 nm8s

typedef void nm8s

Тип характеризует векторные данные как массив 8-ми разрядных чисел со знаком. Начальный адрес массива должен быть выровнен по границе 64р 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 nmtype.h строка 149

7.124.2.20 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 64р слова. Предполагается, что размер массива данного типа как минимум кратен 8.

Диапазон значений:

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

7.124.2.21 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:

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

См. определение в файле nmtype.h строка 411

7.124.2.22 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:

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

См. определение в файле nmtype.h строка 428

7.124.2.23 v16nm16s

 $typedef\ struct\ s\_v16nm16s\ v16nm16s$ 

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

7.124.2.24 v16nm16u

 $typedef\ struct\ s\_v16nm16u\ v16nm16u$ 

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

7.124.2.25 v16nm32s

typedef struct s v16nm32s v16nm32s

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

7.124.2.26 v16nm32u

 $typedef\ struct\ s\_v16nm32u\ v16nm32u$ 

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

7.124.2.27 v16nm4b3u

 $typedef\ v16nm4u\ v16nm4b3u$ 

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

Диапазон значений элементов структуры:

 $[0,\ldots,7]$ 

См. определение в файле nmtype.h строка 1222

7.124.2.28 v16nm4u

 $typedef\ struct\ s\ v16nm4u\ v16nm4u$ 

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

7.124.2.29 v16nm8s

 $typedef\ struct\ s\ v16nm8s\ v16nm8s$ 

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

7.124.2.30 v16nm8s7b

 $typedef\ v16nm8s\ v16nm8s7b$ 

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

Диапазон значений элементов структуры:

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

См. определение в файле nmtype.h строка 1084

7.124.2.31 v16nm8u

 $typedef\ struct\ s\_v16nm8u\ v16nm8u$ 

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

7.124.2.32 v2nm32s  $typedef\ struct\ s\_v2nm32s\ v2nm32s$ Тип векторной структуры, состоящей из 2-х 32р. чисел со знаком. 7.124.2.33 v2nm32u typedef struct s v2nm32u v2nm32u Тип векторной структуры, состоящей из 2-х 32р. чисел без знака. 7.124.2.34 v4nm16s  $typedef\ struct\ s\ v4nm16s\ v4nm16s$ Тип векторной структуры, состоящей из 4-х 16р. чисел со знаком. 7.124.2.35 v4nm16u typedef struct s v4nm16u v4nm16u Тип векторной структуры, состоящей из 4-х 16р. чисел без знака.  $7.124.2.36 \quad v4nm32s$ typedef struct s v4nm32s v4nm32s Тип векторной структуры, состоящей из 4-х 32р. чисел со знаком. 7.124.2.37 v4nm32u  $typedef\ struct\ s\_v4nm32u\ v4nm32u$ Тип векторной структуры, состоящей из 4-х 32р. чисел без знака. 7.124.2.38 v4nm8u  $typedef\ struct\ s\ v4nm8u\ v4nm8u$ Тип векторной структуры, состоящей из 4-х 8р. чисел без знака.

typedef struct s\_v8nm16s v8nm16s

7.124.2.39 v8nm16s

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

 $\Gamma$ руппы

7.124.2.40 v8nm16u

 $typedef\ struct\ s\ v8nm16u\ v8nm16u$ 

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

7.124.2.41 v8nm32s

typedef struct s v8nm32s v8nm32s

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

7.124.2.42 v8nm32u

 $typedef\ struct\ s\_v8nm32u\ v8nm32u$ 

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

7.124.2.43 v8nm8s

 $typedef\ struct\ s\_v8nm8s\ v8nm8s$ 

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

7.124.2.44 v8nm8u

 $typedef\ struct\ s\_v8nm8u\ v8nm8u$ 

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

#### 7.125 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 uint 7b
- 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

#### 7.125.1Подробное 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 низкоурвневых функций. Функции библиотеки имеют С++ 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 х86, выполненных на языке С++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

#### 7.125.2 Типы

7.125.2.1 int15b

typedef int int15b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 596

7.125.2.2 int16b

typedef int int16b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 609

7.125.2.3 int1b

typedef int int1b

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

Диапазон значений:

[-1, 0]

См. определение в файле nmtype.h строка 518

7.125.2.4 int2b

typedef int int2b

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

Диапазон значений:

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

7.125.2.5 int 30b

typedef int int30b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 622

7.125.2.6 int31b

typedef int int31b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 635

7.125.2.7 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 nmtype.h строка 648

7.125.2.8 int3b

typedef int int3b

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

Диапазон значений:

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

7.125.2.9 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 nmtype.h строка 557

7.125.2.10 int 63b

typedef INT64 int63b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 661

7.125.2.11 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 nmtype.h строка 674

7.125.2.12 int7b

typedef int int7b

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

Диапазон значений:

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

7.125.2.13 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 nmtype.h строка 583

7.125.2.14 uint15b

typedef unsigned int uint15b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 765

7.125.2.15 uint16b

typedef unsigned int uint 16b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 778

7.125.2.16 uint1b

typedef unsigned int uint1b

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

Диапазон значений:

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

7.125.2.17 uint2b

typedef unsigned int uint2b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 700

7.125.2.18 uint31b

typedef unsigned int uint31b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 791

7.125.2.19 uint32b

typedef unsigned int uint32b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 804

7.125.2.20 uint3b

typedef unsigned int uint3b

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

Диапазон значений:

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

7.125.2.21 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 nmtype.h строка 726

7.125.2.22 uint63b

typedef UINT64 uint63b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 817

7.125.2.23 uint64b

typedef nm64u uint64b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 830

7.125.2.24 uint7b

typedef unsigned int uint7b

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 739

7.125.2.25 uint8b

typedef unsigned int uint8b

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

Диапазон значений:

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

# 7.126 Функции поддержки

### Группы

• nmppsMalloc

Распределение памяти для векторов библиотеки.

 $\bullet$  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.

### 7.126.1 Подробное описание

## 7.127 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.

### 7.127.1 Подробное описание

214 Группы

## 7.128 Арифметические операции

### Группы

• nmppsAbs

Вычисление абсолютных значений для элементов вектора.

• nmppsAbs1

Функция логического вычесления модулей элементов вектора.

• nmppsNeg

Изменение знака элементов вектора на противоположный.

• nmppsAddC

Добавление к вектору константы.

• nmppsAdd

Сложение двух векторов.

 $\bullet$  nmppsAdd\_AddC

Сложение двух векторов с прибавлением константы.

• nmppsSubC

Вычитание константы из вектора.

• nmppsSubCRev

Вычитание константы из вектора с переменой знака элементов вектора.

• nmppsSub

Вычитание двух вектров.

• 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.

 $\bullet$  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

#### 7.128.1 Подробное описание

### 7.129 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.

 $\bullet \ nmpps Displace Bits$ 

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:

7.129.1 Подробное описание

216 Группы

## 7.130 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.

#### 7.130.1 Подробное описание

# 7.131 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

### 7.131.1 Подробное описание

# 7.132 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)

#### 7.132.1 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 Результи	рующий вектор.
------------------	----------------

#### Возвращает

void

#### Restrictions:

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

7.133 nmppsAbs1 219

# 7.133 nmppsAbs1

Функция логического вычесления 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)

#### 7.133.1 Подробное описание

Функция логического вычесления модулей элементов вектора.

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

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

#### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

Возвращает

 $\Gamma$ руппы

# 7.134 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)

### 7.134.1 Подробное описание

Изменение знака элементов вектора на противоположный.

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

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

#### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

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

Возвращает

7.135 nmppsAddC 221

# 7.135 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)

#### 7.135.1 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

# 7.136 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 src← Step2, nm64s \*pDstVec, int dstStep, int nSize)

#### 7.136.1 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

# $7.137 \text{ nmppsAdd\_AddC}$

Сложение двух векторов с прибавлением константы.

# Функции

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

# 7.137.1 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

#### Возвращает

224 Группы

# 7.138 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)

#### 7.138.1 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

7.139 nmppsSubCRev 225

# 7.139 nmppsSubCRev

Вычитание константы из вектора с переменой 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)

### 7.139.1 Подробное описание

Вычитание константы из вектора с переменой 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

рDstVec Результирующий вектор.
--------------------------------

Возвращает

226 Группы

# 7.140 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)

#### 7.140.1 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

7.141 nmppsAbsDiff 227

# 7.141 nmppsAbsDiff

Вычисление вектора модулей разности элементов двух векторов.

### Функции

- void nmpps AbsDiff\_8s (const nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)
- $\bullet \ \ void \ nmppsAbsDiff\_16s \ (const \ nm16s \ *pSrcVec1, \ nm16s \ *pSrcVec2, \ nm16s \ *pDstVec, \ int \ nSize)$
- void nmppsAbsDiff 32s (const nm32s \*pSrcVec1, nm32s \*pSrcVec2, nm32s \*pDstVec, int nSize)
- void nmppsAbsDiff 64s (const nm64s \*pSrcVec1, nm64s \*pSrcVec2, nm64s \*pDstVec, int nSize)

### 7.141.1 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,  $-2^{15}$  и т.д). В противном случае, абсолютное значение для таких элементов вычисляется не верно, давая на выходе то же самое число.

# 7.142 nmppsAbsDiff1

Функция логического вычисления модулей разностей элементов двух векторов.

### Функции

• void nmppsAbsDiff1 8s (nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)

### 7.142.1 Подробное описание

Функция логического вычисления модулей разностей элементов двух векторов.

$$pDstVec[i] = \{ \begin{array}{ccc} pSrcVec1[i] - pSrcVec2[i], & if & pSrcVec1[i] - pSrcVec2[i] \geq 0 \\ pSrcVec1[i] - pSrcVec2[i] - 1, & if & pSrcVec1[i] - pSrcVec2[i] < 0 \end{array} \}$$

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

#### 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 $^15$  и т.д). В противном случае, абсолютное значение для таких элементов вычисляется не верно, давая на выходе то же самое число.

7.143 nmppsMulC 229

# 7.143 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)

#### 7.143.1 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

 $\Gamma$ руппы

# $7.144 \text{ nmppsMul\_AddC}$

Поэлементное умножение векторов с прибавлением константы.

# Функции

• void nmppsMul\_AddC\_64s (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, const nm64s \*pnVal, nm64s \*pDstVec, int nSize)

# 7.144.1 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

Возвращает

# 7.145 nmppsMulC AddC

Умножение вектора на константу с прибавлением константы.

### Функции

- 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.

#### 7.145.1 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.
nMul $V$ al	Константа-множитель.
nAddVal	Добавляемая константа.
nSize	Размер векторов в элементах.

Возвращаемые значения

#### Возвращает

 $\operatorname{void}$ 

#### 7.145.2 Функции

 $\Gamma$ руппы

### 7.145.2.1 nmppsMulC\_AddC\_2x32s()

```
\label{eq:condition} 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.

$$\begin{aligned} dataSparseDst[i \cdot stepSparseDst][k] &= dataSparseSrc[i \cdot stepSparseSrc][k] \cdot mulArg[k] + addArg[k], \\ i &= \overline{0 \dots size - 1}; k = \overline{0 \dots K - 1}, \end{aligned}$$

where K is value of intWxK type.

#### Аргументы

in	${\rm dataSparseSrc}$	Input sparse vector of 64-bit packed words
in	$\mathrm{mulArg}$	Packed 64-bit word with values to multiply
in	$\operatorname{addArg}$	Packed 64-bit word with values to add
in	${ m dataSparseDst}$	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	$64\mbox{-bit}$ step between input packed words in memory . By default=1 means that input vector is continuous
in	step Sparse Dst	64-bit step between output packed words in memory. By default=1 means that output vector is continuous

Возвращает

# $7.146 \quad nmppsRShiftC\_MulC\_AddC$

# Функции

• void nmppsRShiftC\_MulC\_AddC\_2x32s (int32x2 \*dataSparseSrc, int32x2 \*preshiftArg, int32x2 \*mulArg, int32x2 \*addArg, int32x2 \*dataSparseDst, int size, int stepSparseSrc, int stepSparseDst)

### 7.146.1 Подробное описание

Sparse vector by constant multiplication with addition of constant.

 $dataSparseDst[i\cdot stepSparseDst][k] = (dataSparseSrc[i\cdot stepSparseSrc][k] >> preshiftArg[k]) \cdot mulArg[k] + addArg[k], \\ + addArg[k] + add$ 

$$i = \overline{0 \dots size - 1}; k = \overline{0 \dots K - 1},$$

where K is value of intWxK type

#### Аргументы

in	dataSparseSrc	Input sparse vector of 64-bit packed words
in	$\operatorname{preshiftArg}$	Packed 64-bit word of values for preshifting of input data $= [2,4,6,828,30]$
in	$\mathrm{mulArg}$	Packed 64-bit word with values to multiply
in	$\operatorname{addArg}$	Packed 64-bit word with values to add
in	${ m dataSparseDst}$	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	step Sparse Dst	64-bit step between output packed words in memory. By default=1 means that output vector is continuous

Возвращает

# $7.147 \quad nmppsMulC\_AddV\_AddC$

Умножение вектора на константу с прибавлением вектора и константы.

# Функции

• void nmppsMulC\_AddV\_AddC\_32s (nm32s \*pSrcVec1, int nMulVal, nm32s \*pSrcVec2, int n  $\leftarrow$  AddVal, nm32s \*pDstVec, int nSize)

# 7.147.1 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 Результирун	ющий вектор.
---------------------	--------------

Возвращает

7.148 nmppsSumN

# 7.148 nmppsSumN

Сложение нескольких векторов.

# Функции

- $\bullet \ \ void\ nmppsSumN\_8s16s\ (nm8s\ **ppSrcVec,\ nm16s\ *pDstVec,\ int\ nSize,\ int\ nNumberOfVectors)$
- void nmppsSumN\_16s (nm16s \*\*ppSrcVec, nm16s \*pDstVec, int nSize, int nNumberOfVectors)

# 7.148.1 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

Возвращает

# 7.149 nmppsDivC

Деление вектора на константу.

### Функции

• void nmppsDivC\_32s (nm32s \*pSrcVec, int nDivisor, nm32s \*pDstVec, int nSize, void \*pTmpBuf1, void \*pTmpBuf2)

#### 7.149.1 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 Результир	ующий 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 коректного использования in-place параметров следует учитывать порядок получения промежуточных результатов:

the order of obtaining intermediate results:

 $\begin{array}{l} pSrcVec => pTmpBuf1 \ (1cpl:L <=>G) - 1/x \ Multiplying \ (in-place \ is \ supported) \\ pTmpBuf1 => pTmpBuf2 \ (1cpl:G <=>L) - Scaling \ down \ (in-place \ is \ supported) \end{array}$ 

7.149 nmppsDivC 237

 $pTmpBuf2 => pDstVec \ (2cpl:L <=>G) \ - \ Result \ correction \ (in-place \ is \ supported)$ 

Примеры использования in-place параметров:  $\begin{array}{l} nmppsDiv\_(L0,\!G0,\!10240,\!3,\!G0,\!L0); \\ nmppsDiv\_(L0,\!L0,\!10240,\!3,\!L0,\!L0); \end{array}$ 

# 7.150 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)

#### 7.150.1 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.

7.151 nmppsDotProd 239

# 7.151 nmppsDotProd

Скалярное умножение двух векторов.

#### Функции

- int nmppsDotProd\_8s8sm (const nm8s \*pSrcVec1, const nm8s \*pSrcVec2, int nSize, int64b \*pn← Res, 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)

#### 7.151.1 Подробное описание

Скалярное умножение двух векторов.

$$nRes = \sum_{i=}^{nSize-1} pSrcVec1[i] \cdot pSrcVec2[i]$$

#### 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

240Группы

# 7.152 nmppsWeightedSum

Поэлементное взвешенное суммирование элементов двух векторов

### Функции

- void nmpps WeightedSum\_8s16s (nm8s \*pSrcVec1, int nW1, nm8s \*pSrcVec2, int nW2, nm16s \*pDstVec, int nSize)
- void nmpps WeightedSum\_16s32s (nm16s \*pSrcVec1, int nW1, nm16s \*pSrcVec2, int nW2, nm32s \*pDstVec, int nSize)
- void nmpps WeightedSum\_32s64s (nm32s \*pSrcVec1, nm64s nW1, nm32s \*pSrcVec2, nm64s nW2, nm64s \*pDstVec, int nSize)

#### 7.152.1 Подробное описание

Поэлементное взвешенное суммирование элементов двух векторов

$$\begin{split} pDstVec[i] &= nW1 \cdot pSrcVec1[i] + nW2 \cdot pSrcVec2[i], \\ &= \overline{0 \dots nSize - 1} \end{split}$$

\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

 $\Gamma$ руппы

# 7.153 nmppsNot

Функция логического "НЕ".

# Функции

- 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)

### 7.153.1 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

Возвращает

7.154 nmppsAndC 243

# 7.154 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)

#### 7.154.1 Подробное описание

Функция логического "И" между вектором и константой.

$$pDstVec[i] = pSrcVec[i] \land nVal,$$

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

#### Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

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

Возвращает

# 7.155 nmppsAnd

Функция логического "И" между двумя векторами.

### Функции

- void nmppsAnd 1 (const nm1 \*pSrcVec1, const nm1 \*pSrcVec2, nm1 \*pDstVec, int nSize)
- $\bullet \ \ void \ nmppsAnd \_2u \ (const \ nm2u \ *pSrcVec1, \ const \ nm2u \ *pSrcVec2, \ nm2u \ *pDstVec, \ int \ nSize)$
- 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 nmpps And\_64u (const nm64u \*pSrcVec1, const nm64u \*pSrcVec2, nm64u \*pDstVec, int nSize)

#### 7.155.1 Подробное описание

Функция логического "И" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \land pSrcVec2[i]$$

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

#### Аргументы

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

Возвращаемые значения

pDstVec	Результирующий вектор.
-	

Возвращает

 $7.156~\mathrm{nmppsAnd4V}\_$ 

# 7.156 nmppsAnd4V\_

Функция логического "И" между четырьмя векторами.

# Функции

• void nmppsAnd4V\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pSrcVec3, nm64u \*pSrcVec3, nm64u \*pSrcVec4, nm64u \*pDstVec, int nSize)

# 7.156.1 Подробное описание

Функция логического "И" между четырьмя векторами.

$$pDstVec[i] = pSrcVec1[i] \land pSrcVec2[i] \land pSrcVec3[i] \land pSrcVec4[i]$$

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

#### 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

 $\operatorname{void}$ 

 $\Gamma$ руппы

# 7.157 nmppsAndNotV\_

Функция логического "И-НЕ" между двумя векторами.

# Функции

- void nmppsAndNotV\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pDstVec, int nSize)

# 7.157.1 Подробное описание

Функция логического "И-НЕ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \land notpSrcVec2[i]$$

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

#### Аргументы

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

#### Возвращаемые значения

pDstVec	Результирующий вектор.

Возвращает

7.158 nmppsOrC

# 7.158 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)

### 7.158.1 Подробное описание

Функция логического "ИЛИ" между вектором и константой.

$$pDstVec[i] = pSrcVec[i] \lor nVal$$

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

#### Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.

Возвращает

# 7.159 nmppsOr

Функция логического "ИЛИ" между двумя векторами.

# Функции

- void nmppsOr 1 (const nm1 \*pSrcVec1, const nm1 \*pSrcVec2, nm1 \*pDstVec, int nSize)
- $\bullet \ \ void\ nmppsOr\_2u\ (const\ nm2u\ *pSrcVec1,\ const\ nm2u\ *pSrcVec2,\ nm2u\ *pDstVec,\ int\ nSize)$
- 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)

### 7.159.1 Подробное описание

Функция логического "ИЛИ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \lor pSrcVec2[i]$$

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

#### Аргументы

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

#### Возвращаемые значения

Возвращает

 $7.160~\mathrm{nmppsOr3V}_{\_}$ 

# 7.160 nmppsOr3V

Функция логического "ИЛИ" между четырьмя векторами.

# Функции

• void nmppsOr3V\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pSrcVec3, nm64u \*pDstVec, int nSize)

# 7.160.1 Подробное описание

Функция логического "ИЛИ" между четырьмя векторами.

$$pDstVec[i] = pSrcVec1[i] \lor pSrcVec2[i] \lor pSrcVec3[i]$$

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

#### Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
pSrcVec3	Третий входной вектор.
nSize	Размер векторов в элементах.

### Возвращаемые значения

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

Возвращает

 $\Sigma$  250

# 7.161 nmppsOr4V

Функция логического "ИЛИ" между четырьмя векторами.

# Функции

• void nmppsOr4V\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pSrcVec3, nm64u \*pSrcVec3, nm64u \*pSrcVec4, nm64u \*pDstVec, int nSize)

# 7.161.1 Подробное описание

Функция логического "ИЛИ" между четырьмя векторами.

$$pDstVec[i] = pSrcVec1[i] \lor pSrcVec2[i] \lor pSrcVec3[i] \lor pSrcVec4[i]$$

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

#### 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

7.162 nmppsXorC 251

# 7.162 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)

### 7.162.1 Подробное описание

Функция логического "Исключающего ИЛИ" между вектором и константой.

$$pDstVec[i] = pSrcVec[i] \lor nVal$$

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

#### Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

### Возвращаемые значения

Возвращает

252 Группы

# 7.163 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)

### 7.163.1 Подробное описание

Функция логического "Исключающего ИЛИ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \lor pSrcVec2[i]$$

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

#### Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

|--|

### Возвращает

 $7.164~\mathrm{nmppsMaskV}_{\_}$  253

# 7.164 nmppsMaskV

Функция логического ИЛИ с предварительным маскированием двух векторов.

# Функции

• void nmppsMaskV\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pMaskVec, nm64u \*p  $\hookrightarrow$  DstVec, int nSize)

# 7.164.1 Подробное описание

Функция логического ИЛИ с предварительным маскированием двух векторов.

$$pDstVec[i] = (pSrcVec1[i] \qquad and \qquad pMaskVec[i]) \qquad or \qquad (pSrcVec2[i] \qquad and \qquad \overline{pMaskVec[i]})$$

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

### Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
pMaskVec	Вектор маски.
nSize	Размер векторов в элементах.

# Возвращаемые значения

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

Возвращает

254 Группы

# 7.165 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)

# 7.165.1 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

7.166 nmppsRShiftC 255

# 7.166 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)

# 7.166.1 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

### Возвращает

256 Группы

# 7.167 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 консанты. 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

$\mathrm{pDstVec}$	Результирующий вектор.
--------------------	------------------------

Возвращает

# 7.168 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  $\leftarrow$  Bits, int nSize)

# 7.168.1 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:

```
\begin{array}{l} pnBits = & [AB000000000000000] \\ pSrcVec = & [0807060504030201] [FF0F0E0D0C0B0A09] \\ pDstVec = & [07060504030201AB] [0F0E0D0C0B0A0908] \end{array}
```

Последние 8 бит массива pDstVec будут потеряны. Если же указатель pBits установить на последнее 64p. 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=[2,4,662].	

### Возвращаемые значения

pDstVec	Результирующий вектор.

258 Группы

Возвращает

# 7.169 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\_64up (nm64u \*pVec, uint64b \*nVal, int nSize)

# 7.169.1 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 соответсвовать типу результирующего	
	вектора.	

Возвращаемые значения

p	Vec	Результирующий вектор.

Возвращает

# 7.170 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)
```

## 7.170.1 Подробное описание

Инициализация массива случайными числами.

#### Аргументы

nSize	Размер вектора.
nRandomize	Произвольное число для инициализации генератора случайных чисел.

Возвращаемые значения

p]	$\mathrm{DstVec}$	Результирующий вектор.

Возвращает

void

## 7.170.2 Функции

```
7.170.2.1 nmppsRandUniform 64s()
```

/ru Инициализация массива 32-разрядными случайными числами. /en Random initialization of 32-bit buffer  $/\sim$ 

# 7.171 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 ()

# 7.171.1 Подробное описание

Генерация случайного числа с равномерным распределением.

/ru Инициализация массива 64-разрядными случайными числами. /en Random initialization of 64-bit buffer  $/\sim$ 

# Аргументы

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

## Возвращает

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

# 7.172 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)

# 7.172.1 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

7.173 nmppsConvert 263

#### 7.173 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

## 7.173.1 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

## Возвращает

void

## 7.173.2 Функции

Restrictions: nSize = [32\*64,32\*64\*2,32\*64\*3,...]

7.173 nmppsConvert 265

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

 $\Phi$ ункция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

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

 $\Phi$ ункция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

Функция конвертации вектора целых беззнаковых чисел в вектор комплексных чисел, где мни-мая(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

```
7.173.2.6 \quad 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

```
7.173.2.7 \quad 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

7.173 nmppsConvert 267

```
7.173.2.8 \quad nmppsJoin\_32f() void \ nmppsJoin\_32f \ ( const \ nm32f * pSrcVec1, const \ nm32f * pSrcVec2, nm32f * pDstVec, int \ nSize \ )
```

 $\Phi$ ункция соединяет 2 массива 32-х чисел с плавающей точкой (float) в один. Результирующий массив { pSrcVec1[0], pSrcVec2[0], pSrcVec1[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)

# 7.174 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)

# 7.174.1 Подробное описание

Копирование вектора.

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

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

## Аргументы

$\mathrm{pSrcVec}$	Входной вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

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

Возвращает

 $7.175 \text{ nmppsCopyua}_{\_}$  269

# 7.175 nmppsCopyua

Копирование вектора с невыровненной байтовой позиции в выровненную.

# Функции

• void nmppsCopyua\_8s (const nm8s \*pSrcVec, int nSrcOffset, nm8s \*pDstVec, int nSize)

# 7.175.1 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.
${ m nSrcOffset}$	Смещение в элементах относительно начала вектора. nSrcOffset Может принимать
	любое значение.
nSize	Кол-во копируемых элементов.

Возвращает

# 7.176 nmppsSwap\_

 $\Pi$ ерестановка двух векторов.

# Функции

• void nmppsSwap\_64s (nm64s \*pSrcVec1, nm64s \*pSrcVec2, int nSize)

# 7.176.1 Подробное описание

Перестановка двух векторов.

# Аргументы

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

Возвращает

7.177 nmppsMax 271

# 7.177 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)

## 7.177.1 Подробное описание

Поиск значения максимального элемента вектора.

$$nMaxValue = \max_{i} (pSrcVec[i])$$

#### Аргументы

$\mathrm{pSrcVec}$	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.

# 7.177.2 Функции

#### Restrictions:

Максимальный и минимальный элементы массива должны отличатся не более чем на  $2^{\wedge}15$ -1. Примеры допустимых диапазонов входных чисел:

#### Restrictions:

Максимальный и минимальный элементы массива должны отличатся не более чем на  $2^31-1$ . Примеры допустимых диапазонов входных чисел:  $[00000000h..7FFFFFFFh] = [0..+2^31-1] [FFFFFFFh..7FFFFFFh] = [-1..+2^31-2] [C0000000h..3 \leftarrow FFFFFFFh] = [-2^30..+2^30-1] [80000000h..00000000h] = [-2^31..0]$ 

#### Restrictions:

Физический размер вектора должен быть кратен блоку из 32-х 64р. слов

Максимальный и минимальный элементы массива должны отличатся 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]

7.178 nmppsMin 273

# 7.178 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)

## 7.178.1 Подробное описание

Поиск значения минимального элемента вектора.

$$nMinValue = \min_{i} (pSrcVec[i])$$

#### Аргументы

$\operatorname{pSrcVec}$	Входной вектор.	
nSize	Размер вектора в элементах.	

## Возвращаемые значения

3 51 77 1			
${ m nMinValue}$	значение	минимального	элемент.

Возвращает

void

## Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

# 7.178.2 Функции

#### Restrictions:

Максимальный и минимальный элементы массива должны отличатся не более чем на  $2^{\wedge}15$ -1. Примеры допустимых диапазонов входных чисел:

```
 \begin{array}{l} [0000h..7FFFh] = [ \ 0.. + 32767 \ ] \ [FFFFh..7FFEh] = [ \ -1.. + 32766 \ ] \ [C000h..3FFFh] = [ \ -16384.. + 16383 \ ] \\ [8000h..0000h] = [ \ -32768..0 \ ] \end{array}
```

```
7.178.2.2 \quad nmppsMin\_32s31b() void \ nmppsMin\_32s31b \ ( const \ nm32s31b \ * pSrcVec, int \ nSize, int \ * nMinValue \ )
```

#### Restrictions:

Максимальный и минимальный элементы массива должны отличатся не более чем на  $2^{\wedge}31$ -1. Примеры допустимых диапазонов входных чисел:

```
 \begin{array}{l} [00000000h..7FFFFFFFh] = & [\ 0...+2^31-1] \ [FFFFFFFh..7FFFFFEh] = & [\ -1...+2^31-2\ ] \ [C0000000h..3 \longleftrightarrow FFFFFFFh] = & [\ -2^30...+2^30-1\ ] \ [80000000h..00000000h] = & [\ -2^31..0\ ] \end{array}
```

## Restrictions:

Максимальный и минимальный элементы массива должны отличатся 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]
```

# 7.179 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 \*p↔ LTmpBuf, 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)

### 7.179.1 Подробное описание

Поиск значения максимального элемента вектора и его положения (положений) в векторе.

$$nMaxValue = \max_{i} \left( pSrcVec[i] \right)$$

### 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.
111114011	Tingene nepbere nangemere makenaj ma epega pablisa.

## 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 отличатся 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:  $\n$ 

 $\begin{array}{l} [0000h..7FFFh] = [ \ 0... + 32767 \ ] \ [FFFFh..7FFEh] = [ \ -1... + 32766 \ ] \ [C000h..3FFFh] = [ -16384... + 16383 \ ] \ [8000h..0000h] = [ -32768..0 \ ] \end{array}$ 

# 7.180 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 \*pL← TmpBuf, 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)

### 7.180.1 Подробное описание

Поиск значения минимального элемента вектора и его положения (положений) в векторе.

$$nMinValue = \min_{i} \left( pSrcVec[i] \right)$$

## 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

ndex 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 отличатся 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:  $\n$ 

 $\begin{array}{l} [0000h..7FFFh] = [ \ 0.. + 32767 \ ] \ [FFFFh..7FFEh] = [ \ -1.. + 32766 \ ] \ [C000h..3FFFh] = [ -16384.. + 16383 \ ] \\ [8000h..0000h] = [ -32768..0 \ ] \end{array}$ 

# 7.181 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)

## 7.181.1 Подробное описание

Поиск значения минимального элемента вектора длины N и его положения в векторе.

### Аргументы

pSrc	·Vec	Массив из N элементов, где N определяется числом в имени функции.
nStr	ride	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.

# 7.182 nmppsFirstZeroIndx

Поиск позиции первого нулевого элемента в векторе .

# Функции

• int nmppsFirstZeroIndx\_32s (int \*pSrcVec, int nSize)

# 7.182.1 Подробное описание

Поиск позиции первого нулевого элемента в векторе.

## Аргументы

pSrcVec	Входной массив.
nIndex	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.

# 7.183 nmppsFirstNonZeroIndx

Поиск позиции первого ненулевого элемента в векторе .

# Функции

• int nmppsFirstNonZeroIndx\_32s (int \*pSrcVec, int nSize)

# 7.183.1 Подробное описание

Поиск позиции первого ненулевого элемента в векторе .

## Аргументы

pSrcVec	Входной массив.
nIndex	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.

# $7.184 \quad nmppsLastZeroIndx$

Поиск позиции последнего нулевого элемента в векторе.

# Функции

• int nmppsLastZeroIndx\_32s (int \*pSrcVec, int nSize)

# 7.184.1 Подробное описание

Поиск позиции последнего нулевого элемента в векторе.

## Аргументы

pSrcVec	Входной массив.
nIndex	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.

# 7.185 nmppsLastNonZeroIndx

Поиск позиции последнего ненулевого элемента в векторе.

# Функции

- int nmppsLastNonZeroIndx\_32s (int \*pSrcVec, int nSize)

# 7.185.1 Подробное описание

Поиск позиции последнего ненулевого элемента в векторе .

## Аргументы

pSrcVec	Входной массив.
nIndex	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 ненулевого элемента.

284 Группы

# 7.186 nmppsMinEvery

Поэлементный минимум из двух векторов.

# Функции

- $\bullet \ \ void \ nmppsMinEvery \ \ 8s \ (nm8s7b \ *pSrcVec1, \ nm8s7b \ *pSrcVec2, \ nm8s7b \ *pDstMinVec, \ int \ nSize) \\$
- 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)

# 7.186.1 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 отличатся 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:

 $\begin{array}{c} [0000h..7FFFh] = [ \ 0.. + 32767 \ ] \\ [FFFFh..7FFEh] = [ \ -1.. + 32766 \ ] \end{array}$ 

[C000h..3FFFh]=[-16384..+16383]

[8000h..0000h]=[-32768..0]

# 7.187 nmppsMaxEvery

Поэлементный максимум из двух векторов.

# Функции

- void nmppsMaxEvery\_8s (nm8s7b \*pSrcVec1, nm8s7b \*pSrcVec2, nm8s7b \*pDstMaxVec, int n  $\hookleftarrow$  Size)
- 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)

## 7.187.1 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 отличатся 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]

286 Группы

# 7.188 nmppsMinCmpLtV

Поэлементный минимум из двух векторов.

# Функции

• void nmppsMinCmpLtV\_16s (nm16s15b \*pSrcVec1, nm16s15b \*pSrcVec2, nm16s15b \*pDstMin, nm16s15b \*pDstSignMask, int nSize)

## 7.188.1 Подробное описание

Поэлементный минимум из двух векторов.

$$pDstMin[i] = min(pSrcVec1[i], pSrcVec2[i])$$

$$pDstSignMask[i] = \left\{ \begin{array}{ll} 11 \dots 1b, & pSrcVec1[i] < pSrcVec2[i] \\ 00 \dots 0b, & pSrcVec1[i] <= pSrcVec2[i] \end{array} \right.$$

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

## 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 отличатся 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:

$$\begin{array}{l} [0000h..7FFFh] = [ \ 0.. + 32767 \ ] \\ [FFFFh..7FFEh] = [ \ -1.. + 32766 \ ] \\ [C000h..3FFFh] = [ \ -16384.. + 16383 \ ] \\ [8000h..0000h] = [ \ -32768..0 \ ] \end{array}$$

288 Группы 7.189 nmppsCmpLt0

## 7.189 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)

## 7.189.1 Подробное описание

Сравнивает элементы массива на меньше нуля.

$$pDstVec(i) = \{ \begin{array}{lll} 11 \dots 1b, & if & pSrcVec(i) < 0 \\ 00 \dots 0b, & if & pSrcVec(i) \geq 0 \end{array}$$

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

Если элемент входного вектора меньше 0, во все биты соответствующего элемента выходного вектора записывается 1.

#### Аргументы

$\mathrm{pSrcVec}$	Входной вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

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

Возвращает

## 7.190 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)

### 7.190.1 Подробное описание

Сравнивает элементы массива на признак равенства нулю.

$$pDstVec(i) = \{ \begin{array}{ll} 1, & if & pSrcVec(i) = 0 \\ 0, & if & pSrcVec(i) \neq 0 \end{array}, if \ nTrueFlag = 1$$

$$pDstVec(i) = \{ \begin{array}{ll} 0, & if & pSrcVec(i) = 0 \\ 1, & if & pSrcVec(i) \neq 0 \end{array}, if \ nTrueFlag = 0$$

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

#### Аргументы

pSrcVec	Входной вектор.	
nSize	Размер векторов в элементах. Vector size in elements.	
nTrueFlag	0 1 0	
	условия.	

#### Возвращаемые значения

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

#### Возвращает

void

#### 7.190.2 Функции

7.190 nmppsCmpEq0

```
7.190.2.1 nmppsCmpEq0_16u15b()
void nmppsCmpEq0\_16u15b (
             nm16u15b*pSrcVec,\\
             nm1*\mathrm{pDstVec},
              int nSize,
              int nTrueFlag )
Заметки
     nSize = [1,2,3,4...]
7.190.2.2 nmppsCmpEq0_32u31b()
void nmppsCmpEq0\_32u31b (
             nm32u31b * pSrcVec,
             nm1 * pDstVec,
             int nSize,
             int nTrueFlag)
Заметки
     nSize = [1,2,3,4...]
7.190.2.3 nmppsCmpEq0_8u7b()
void nmppsCmpEq0\_8u7b (
             nm8u7b*pSrcVec,\\
             nm1*\mathrm{pDstVec},
             int nSize,
             int nTrueFlag )
Заметки
      nSize = [1,2,3,4...]
```

# $7.191 \quad nmppsCmpMinMaxV\_$

Поэлементное сравнение двух векторов.

### Функции

- void nmppsCmpMinMaxV\_8s (nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstMin, nm8s \*pDst $\leftarrow$  Max, 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)

#### 7.191.1 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.
${ m pSrcVec2}$	Второй Входной вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

pDstMin	Массив поэлементных минимумов для входных массивов.	Ī
pDstMax	Массив поэлементных максимумов для входных массивов.	]

#### Возвращает

void

#### Restrictions:

Сравниваемые пары чисел двух массивов должны отличатся 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]

 $\begin{array}{l} [{\rm FFFFh..7FFEh}] {=} [ \ \hbox{--}1... {+}32766 \ ] \\ [{\rm C000h..3FFFh}] {=} [\hbox{--}16384... {+}16383 \ ] \\ [{\rm 8000h..0000h}] {=} [\hbox{--}32768..0 \ ] \end{array}$ 

# 7.192 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)

## 7.192.1 Подробное описание

Функция насыщения.

$$pDstVec[i] = \begin{cases} -2^{nClipFactor}, & if \quad pSrcVec[i] < -2^{nClipFactor} \\ pSrcVec[i], & if \quad -2^{nClipFactor} \leq pSrcVec[i] \leq 2^{nClipFactor} - 1 \\ 2^{nClipFactor} - 1, & if \quad pSrcVec[i] > 2^{nClipFactor} - 1 \end{cases}$$

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

#### Аргументы

pSrcVec	Входной вектор.
nClipFactor	Показатель степени, определяющий верхний и нижний пороги насыщения.
nSize	Размер вектора в элементах.

#### Возвращаемые значения

$\operatorname{pDstVec}$	Результирующий вектор.

#### Возвращает

 $7.193~{\rm nmppsClipCC}\_$ 

# 7.193 nmppsClipCC

Функция насыщения с произвольными порогами.

## Функции

• void nmppsClipCC\_32s (nm32s30b \*pSrcVec, int30b nNegThresh, int30b nPosThresh, nm32s30b \*pDstVec, int nSize)

### 7.193.1 Подробное описание

Функция насыщения с произвольными порогами.

$$pDstVec[i] = \left\{ \begin{array}{ll} nPosThresh, & if & pSrcVec[i] > nPosThresh \\ pSrcVec[i], & if & nNegThresh \leq pSrcVec[i] \leq nPosThresh \\ nNegThresh, & if & pSrcVec[i] < nNegThresh \end{array} \right.$$

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

#### 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

296 Группы

# 7.194 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)

### 7.194.1 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. =[1,2,315]
nShift	Параметр определяет на сколько позиций =[2,4,614] нужно сдвинуть биты элементов вектора;
nAddValue	Добавляемая константа.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

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

#### Возвращает

# 7.195 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)

### 7.195.1 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.
	=[1,2,315]
nAddValue	Добавляемая константа.
nSize	Размер векторов в элементах. указатель на матрицу коэфициентов для векторного умножителя. Для ускорения.

#### Возвращаемые значения

pDstVec	Результирующий вектор.

298 Группы

Возвращает

7.196 nmppsCmpEqC 299

## 7.196 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)

#### 7.196.1 Подробное описание

Сравнивает элементы массива на признак равенства константе.

$$pDstVec(i) = \{ \begin{array}{ccc} nTrueFlag, & if & pSrcVec(i) = nCmpVal \\ 0, & if & pSrcVec(i) \neq nCmpVal \end{array}$$

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

#### Аргументы

pSrcVec	Входной вектор.
nCmpVal	Значение константы для сравнияе
nSize	Размер векторов в элементах.
nTrueFlag	Значние флага, устанавливаемого при выполнении условия

## Возвращаемые значения

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

#### Возвращает

## 7.197 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)

## 7.197.1 Подробное описание

Сравнивает элементы массива на признак неравенства нулю.

$$pDstVec(i) = \{ \begin{array}{ccc} nTrueFlag, & if & pSrcVec(i) \neq 0 \\ 0, & if & pSrcVec(i) = 0 \end{array}$$

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

#### Аргументы

pSrcVec	Входной вектор.
nCmpVal	Значение константы для сравнияе
nSize	Размер векторов в элементах.
nTrueFlag	Значние флага, устанавливаемого при выполнении условия

#### Возвращаемые значения

pDstVec	Результирующий вектор.

## Возвращает

7.198 nmppsCmpNeC 301

## 7.198 nmppsCmpNeC

Сравнивает элементы массива на признак неравенства константе.

#### Группы

• nmppsCmpEqV

Поэлементное сравнение элементов двух вектров на признак равенства.

• nmppsCmpNeV

Поэлементное сравнение элементов двух вектров на признак неравенства.

### Функции

- void nmppsCmpNeC 8s (const nm8s \*pSrcVec, int8b nCmpVal, nm8s \*pDstVec, int nSize)
- $\bullet \ \ void\ nmppsCmpNeC\_16s\ (const\ nm16s\ *pSrcVec,\ int16b\ nCmpVal,\ nm16s\ *pDstVec,\ int\ nSize)$
- 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)

#### 7.198.1 Подробное описание

Сравнивает элементы массива на признак неравенства константе.

$$pDstVec(i) = \{ \begin{array}{ccc} nTrueFlag, & if & pSrcVec(i) \neq nCmpVal \\ 0, & if & pSrcVec(i) = nCmpVal \end{array}$$

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

#### Аргументы

pSrcVec	Входной вектор.
nCmpVal	Значение константы для сравнияе
nSize	Размер векторов в элементах.
nTrueFlag	Значние флага, устанавливаемого при выполнении условия

7.198 nmppsCmpNeC 303

Возвращаемые значения

Возвращает

304 Группы

# 7.199 nmppsCmpEqV\_

Поэлементное сравнение элементов двух вектров 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)

## 7.199.1 Подробное описание

Поэлементное сравнение элементов двух вектров на признак равенства.

$$pDstVec(i) = \{ \begin{array}{ccc} nTrueFlag, & if & pSrcVec(i) = nCmpVal \\ 0, & if & pSrcVec(i) \neq nCmpVal \end{array}$$

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

#### Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.
nTrueFlag	Значние флага, устанавливаемого при выполнении условия

### Возвращаемые значения

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

Возвращает

# 7.200 nmppsCmpNeV

Поэлементное сравнение элементов двух вектров на признак неравенства.

## Функции

- void nmppsCmpNeV\_16u (nm16u15b \*pSrcVec1, nm16u15b \*pSrcVec2, nm16s \*pDstVec, int n⇔ Size, int16b nTrueFlag)
- void nmppsCmpNeV\_8u (nm8u7b \*pSrcVec1, nm8u7b \*pSrcVec2, nm8s \*pDstVec, int nSize, int8b nTrueFlag)

## 7.200.1 Подробное описание

Поэлементное сравнение элементов двух вектров на признак неравенства.

$$pDstVec(i) = \{ \begin{array}{ccc} nTrueFlag, & if & pSrcVec(i) \neq nCmpVal \\ 0, & if & pSrcVec(i) = nCmpVal \end{array}$$

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

#### Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.
nTrueFlag	Значние флага, устанавливаемого при выполнении условия

### Возвращаемые значения

рDstVec   Результирующий вектор.
----------------------------------

Возвращает

## 7.201 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__ nm32u * nmppsAddr_32u (const nm32u *pVec, int nIndex)
__INLINE__ nm64u * nmppsAddr_32u (const nm64u *pVec, int nIndex)
__INLINE__ nm64u * nmppsAddr_32u (const nm64u *pVec, int nIndex)
```

#### 7.201.1 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.

7.202 nmppsSetVal 307

## 7.202 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\_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)

#### 7.202.1 Подробное описание

Модификация элемента вектора.

pVec(nIndex) = Val

#### Аргументы

pVec	Вектор.
nIndex	Позиция элемента
nVal	Значение элемента

Возвращает

## 7.203 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)

#### 7.203.1 Подробное описание

Извлекает значение элемента вектора.

#### Аргументы

pVec	Вектор.
nIndex	Позиция элемента.

#### Возвращаемые значения

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

Возвращает

# 7.204 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)
- 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)

### 7.204.1 Подробное описание

Извлекает значение элемента вектора.

#### Аргументы

pVec	Вектор.
nIndex	Позиция элемента.

#### Возвращает

Значение элемента.

# 7.205 VEC\_QSort

Сортировка массива по убыванию.

## Функции

• void nmppsQSort\_32s (nm32s \*pSrcDstVec, int nSize)

## 7.205.1 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\*log2(nSize) 32-битных слов в лучшем случае (элементы массива расположены беспорядочно) и больше 6\*nSize 32-битных слов в худшем случае (элементы массива уже упорядочены)

7.206 nmppsRemap 311

## 7.206 nmppsRemap

Переупорядочивание элементов вектора по таблице.

## Функции

- void nmppsRemap\_32u (nm32u \*pSrcVec, nm32u \*pDstVec, nm32s \*pRemapTable, int nDst $\hookleftarrow$  VecSize)
- void nmppsRemap\_8u (nm8u \*pSrcVec, nm8u \*pDstVec, nm32s \*pRemapTable, int nSrcVecSize, int nDstVecSize, void \*pTmpBuf1, void \*pTmpBuf2)

#### 7.206.1 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 следющие 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 following sequence of actions :

• pSrcVec => pTmpBuf1 - inplace запрещен;

- pTmpBuf1=>pTmpBuf2 inplace запрещен;
- pTmpBuf2=>pDstVec inplace разрешен;

7.207 nmppSplitTmp 313

# 7.207 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	${ m tmpSizeOfDst}$	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 находится 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)

# 7.208 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 находится 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)

7.209 nmppMerge 315

# 7.209 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

# 7.210 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 находится в одном банке. Макс производительность на 64-р. слово результата =1 такт

7.211 nmppsDecimate 317

# 7.211 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 соответстовать длинному 64-р. слову.
		1 0
out	pDst	Выходной массив
in	size	Размер исходного массива в элементах. Кратность параметра size должна соответстовать длинному 64-р. слову.

## Возвращает

Details Максимальная производительность достигается при размещении входных, выходных массивов в разных банках памяти.

# 7.212 Типы данных

# Группы

- Типы векторных данных
- Типы скалярных данных

## 7.212.1 Подробное описание

# 7.213 Векторные функции

## Группы

- Элементарные математические функции
- Функции поддержки
- Инициализация и копирование
- Арифметические операции
- Логические и бинарные операции
- Операции сравнения
- Переупорядочивание и сортировка

## 7.213.1 Подробное описание

# 7.214 Матричные функции

# Группы

- Инициализация и копирование
- Функции поддержки
- Векторно-матричные операции

## 7.214.1 Подробное описание

# 7.215 Функции обработки сигналов

# Группы

- Свертка
- Масочная фильтрация
- Изменение размеров
- Быстрое преобразование Фурье

# 7.215.1 Подробное описание

# 7.216 Функции обработки изображений

## Группы

#### • Floodfill

Исполняет разделение бинарной картинки на односвязные области. Пример вызова:  $no=VL\_ \leftarrow FloodFill32b(pSrcImage, Tetr,Image, pTmpBuff, nSrcWidth, nSrcHeight);$ .

- Переупорядочивание изображений
- Арифметические действия
- Масочная фильтрация
- Инициализация и копирование
- Функции поддержки
- Функции графического вывода текста

## 7.216.1 Подробное описание

## 7.217 Скалярные функции

### Группы

- Инициализация
- Integer operations
- Fix-point 64
- Fix-point 32
- Арифметические операции
- Функции деинтерлейсинга

#### 7.217.1 Подробное описание

#### 7.217.1.1 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 х86, 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 кристале 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 низкоурвневых 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 х86, 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.

# 7.218 Базовые регистровые функции библиотеки

## Группы

- Элементарные функции
- функции взвешенного суммирования
- Целевые функции

## 7.218.1 Подробное описание

## 7.219 контроль переполнения

#### Классы

```
 \begin{array}{l} \bullet \  \, {\rm class} \  \, {\rm nmmtr} \! < T > \\ \bullet \  \, {\rm class} \  \, {\rm nmvecpack} \! < T > \\ \bullet \  \, {\rm class} \  \, {\rm vec} \! < T > \\ \end{array}
```

#### Макросы

• #define GetVec getvec

#### Функции

```
• __INLINE__ ostream & operator<< (ostream &s, mtr< unsigned char > &mtr)
```

#### 7.219.1 Подробное описание

определяет классы, предназначенные для контроля переполнения при реализации библиотеки на PC.

-реализации библиотеки на PC 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.

#### 7.219.2 Макросы

```
7.219.2.1 GetVec
```

#define GetVec getvec

Класс матриц.

#### Примеры:

```
\begin{split} &\inf \; \mathrm{Test}[10] {=} \{1,125,3,4,5,6,7,8,9,10\}; \\ &\inf \; \mathrm{Res} \; [10]; \\ &mt {<} \inf {>} \; \mathrm{AA0}(3,3); \\ &vec {<} \inf {>} \; \mathrm{A0}(3); \\ &scalar {<} \inf {>} \; \mathrm{a0}(2); \\ &mt {<} \inf {>} \; \mathrm{CC0}(3,3); \\ &vec {<} \inf {>} \; \mathrm{CC0}(3,3); \\ &vec {<} \inf {>} \; \mathrm{CC0}(3); \\ &BB0.\mathrm{SetData}(\mathrm{Test}); \\ &BB0{=} \; \mathrm{AA0}; \\ &A0 {=} \; \mathrm{AA0}[2]; \\ &a0 {=} \; \mathrm{AA0}[2][2]; \\ &\mathrm{CC0} {=} \; \mathrm{AA0} {+} \; \mathrm{BB0}; \\ &\mathrm{CC0} {=} \; \mathrm{AA0} {*} \; \mathrm{a0}; \\ &\mathrm{CC0} {=} \; \mathrm{AA0} {*} \; \mathrm{A0}; \\ &\mathrm{CC0} {=} \; \mathrm{AA0} {*} \; \mathrm{BB0}; \\ &\mathrm{CC0} {=} \; \mathrm{AO} {*} \; \mathrm{BB0}; \\ \end{aligned}
```

См. определение в файле tmatrix.h строка 60

## 7.219.3 Функции

См. определение в файле nmtlio.h строка 64

## 7.220 Элементарные функции

## Группы

- $Vec_0_{sub_data}$
- $\bullet \ \ Vec\_activate\_data$
- Vec activate data add 0
- $\bullet \ \ Vec\_activate\_data\_xor\_data$
- $Vec_activate_data_add_ram$
- Vec afifo
- $Vec\_data$
- $\bullet \ \ Vec\_data\_add\_ram$
- Vec\_data\_and\_ram
- $\bullet \ \ Vec\_data\_or\_ram$
- $\bullet \ \operatorname{Vec\_data\_sub\_ram}$
- $Vec_data_xor_ram$
- Vec And
- Vec Mask
- $\bullet \ \ Vec\_Or$
- Vec\_Xor
- Vec Add
- $Vec\_Sub$
- $Vec_not_data$
- $\bullet \ \ Vec\_ram$
- $\bullet \ \ Vec\_ram\_sub\_data$
- Vec vsum activate data 0

## 7.220.1 Подробное описание

#### 7.221 функции взвешенного суммирования

## Группы

- Vec ClipMul2D2W8 AddVr
- $Vec\_ClipMulNDNW2\_AddVr$
- $\bullet$  Vec ClipMulNDNW4 AddVr
- $\bullet \ \ Vec\_ClipMulNDNW8\_AddVr$
- $\bullet$  Vec Mul2D2W2 AddVr
- $\bullet$  Vec Mul2D2W4 AddVr
- Vec Mul2D2W8 AddVr
- $Vec_Mul3D3W2_AddVr$
- $\bullet$  Vec Mul4D4W2 AddVr

- $Vec_vsum_data_0$
- Vec vsum data vr
- $Vec_vsum_shift_data_0$
- $Vec_vsum_shift_data_vr$
- Vec vsum shift data afifo

#### 7.221.1 Подробное описание

## 7.222 Целевые функции

#### Группы

- Vec Add VV shift
- Vec data add afifo
- $\bullet \ \ Vec\_FilterCoreRow2$
- Vec FilterCoreRow4
- $\bullet \ \ Vec\_FilterCoreRow8$
- Vec\_Abs
- Vec ClipExt
- Vec IncNeg
- Vec SubAbs
- Vec\_SubVN\_Abs
- Vec Swap
- $Vec_MUL_2V4toW8_shift$
- $\bullet \ \ Vec\_MUL\_2V8toW16\_shift$
- 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:

 $\bullet \ \ Vec\_DupValueInVector16$ 

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

• Vec BuildDiagWeights8

Построение диагональной матрицы весовых коеффициентов (8x8).

• Vec BuildDiagWeights16

Построение диагональной матрицы весовых коеффициентов (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:

#### 7.222.1 Подробное описание

# $7.223 \quad Vec\_0\_sub\_data$

## Функции

• void vec 0 sub data (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.223.1 Подробное описание

Ядро функции nmppsNeg().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{lll} \text{rep N data} = & [\text{ar0} + + \text{gr0}] \text{ with 0-data;} \\ \text{rep N } & [\text{ar6} + + \text{gr6}] = & \text{afifo;} \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.224 Vec\_activate\_data

## Функции

• void vec activate data (nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.224.1 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 = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	$\operatorname{DstMtr}$ stride

#### Restrictions:

# $7.225 \quad \text{Vec} \_ \text{activate} \_ \text{data} \_ \text{add} \_ 0$

## Функции

• void vec\_activate\_data\_add\_0 (nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.225.1 Подробное описание

Функция производит арифметическую активацию.

Ядро функции nmppsClipPowC\_().

Действие функции эквивалентные следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with activate data + 0; rep N [ar6++gr6]=afifo;
```

#### Аргументы

f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.226 \quad Vec\_activate\_data\_xor\_data$

## Функции

• void vec\_activate\_data\_xor\_data (nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.226.1 Подробное описание

Функция позволяет вычислить приближенное значение модуля.

Ядро функции nmppsAbs1().

Действие функции эквивалентные следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with activate data xor data; rep N [ar6++gr6]=afifo;
```

#### Аргументы

f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.227 Vec\_activate\_data\_add\_ram

## Функции

• void vec\_activate\_data\_add\_ram (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.227.1 Подробное описание

Функция выполняет арифметическую активацию с прибавлением константы.

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} rep\ N\ ram = [ar1] \\ rep\ N\ data = [ar0++gr0]\ with\ activate\ data + ram; \\ rep\ N\ [ar6++gr6] = afifo; \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на 64р. слово
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.228 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)

#### 7.228.1 Подробное описание

Функция служит для суммирования двух массивов со сдвигом результата на 1 бит вправо.

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N ram} = [ar4]; \\ \text{rep N data} = [ar0++gr0] \text{ with data} + 0; \\ \text{rep N data} = [ar1++gr1] \text{ with data} + \text{afifo}; \\ \text{rep N with mask ram,shift afifo,0}; \\ \text{rep N } [ar6++gr6] = \text{afifo}; \\ \end{array}
```

#### 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 = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar6,gr2,gr5.

 $7.229 \text{ Vec\_afifo}$  337

# 7.229 Vec\_afifo

## Функции

• void vec afifo (nmreg ar0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.229.1 Подробное описание

Функция служит для заполнения массива константой.

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} {\rm rep~1~data}{=}[{\rm ar0}]~{\rm with~data}; \\ {\rm rep~1*N~[ar6++gr6]}{=}\,{\rm afifo~with~afifo}; \end{array}
```

#### Аргументы

ar0	указатель на 64р. слово
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.230 \ \mathrm{Vec\_data}$

## Функции

• void vec data (nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.230.1 Подробное описание

Ядро функции nmppsCopy\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N data=}[\text{ar0++gr0}] \text{ with data;} \\ \text{rep N }[\text{ar6++gr6}]\text{=}\text{afifo;} \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.231 \quad Vec\_data\_add\_afifo$

## Функции

• void vec data add afifo (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

## 7.231.1 Подробное описание

Ядро функции nmppsSum().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\label{eq:continuous_section} \begin{array}{ll} rep \ 1 & data = vfalse;\\ .repeat \ N;\\ rep \ 1 & data = [ar0++gr0] \ with \ data + \ afifo; \ (rep1 \ N \ times)\\ .endrepeat;\\ rep \ 1 & [ar6] = afifo; \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на 64р. слово

#### Restrictions:

340 Группы

# $7.232 \quad Vec\_data\_add\_ram$

## Функции

• void vec\_data\_add\_ram (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.232.1 Подробное описание

Ядро функции nmppsAddC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} rep\ N\ ram=[ar1];\\ rep\ N\ data=[ar0++gr0]\ with\ data+ram;\\ rep\ N\ [ar6++gr6]=afifo; \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr stride
ar1	указатаель на 64р. слово-константу
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.233 Vec\_data\_and\_ram

## Функции

• void vec\_data\_and\_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.233.1 Подробное описание

Ядро функции  $nmppsAndC_{-}()$ .

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N ram =}[ar1]; \\ \text{rep N data=}[ar0++gr0] \text{ with data and ram}; \\ \text{rep N }[ar6++gr6]=\text{afifo}; \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатаель на 64р. слово-константу
gr5	Высота матриц N $= [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

 $\Gamma$ руппы

# $7.234 \quad Vec\_data\_or\_ram$

## Функции

• void vec data or ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.234.1 Подробное описание

Ядро функции nmppsOrC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} rep\ N\ ram\ = [ar1];\\ rep\ N\ data = [ar0++gr0]\ with\ data\ or\ ram;\\ rep\ N\ [ar6++gr6] = afifo; \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатаель на 64р. слово-константу
$\operatorname{gr}5$	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.235 \quad Vec\_data\_sub\_ram$

## Функции

• void vec\_data\_sub\_ram (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.235.1 Подробное описание

Ядро функции nmppsSubC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N ram=[ar1];} \\ \text{rep N data=[ar0++gr0] with data-ram;} \\ \text{rep N [ar6++gr6]=afifo;} \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатаель на 64р. слово-константу
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

 $\Gamma$ руппы

# $7.236 \quad \text{Vec\_data\_xor\_ram}$

## Функции

• void vec data xor ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.236.1 Подробное описание

Ядро функции nmppsXorC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} rep \ N \ ram = [ar1]; \\ rep \ N \ data = [ar0++gr0] \ with \ data \ xor \ ram; \\ rep \ N \ [ar6++gr6] = afifo; \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на 64р. слово-константу
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.237 \quad \text{Vec\_FilterCoreRow2}$

## Функции

• void vec\_FilterCoreRow2 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

## 7.237.1 Подробное описание

# 7.238 Vec\_FilterCoreRow4

# Функции

• void vec\_FilterCoreRow4 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

7.238.1 Подробное описание

# $7.239 \quad Vec\_FilterCoreRow8$

## Функции

• void vec\_FilterCoreRow8 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

## 7.239.1 Подробное описание

# 7.240 Vec\_And

## Функции

• void vec And (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.240.1 Подробное описание

Ядро функции  $nmppsAnd_{-}()$ .

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N data=}[ar0++gr0] \text{ with data;} \\ \text{rep N data=}[ar1++gr1] \text{ with data and afifo;} \\ \text{rep N }[ar6++gr6] = \text{afifo;} \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	${ m SrcMtr1\ stride}$
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

 $7.241~{
m Vec\_Mask}$  349

# $7.241 \quad \mathrm{Vec}\_\mathrm{Mask}$

## Функции

• void vec\_Mask (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar2, nmreg gr2, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.241.1 Подробное описание

Ядро функции nmppsMaskV\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N ram =} [ar0++gr0];\\ \text{rep N data=} [ar1++gr1] & \text{with data};\\ \text{rep N data=} [ar2++gr2] & \text{with mask data,ram,afifo};\\ \text{rep N } [ar6++gr6] = \text{afifo}; \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar2	указатель на столбец SrcMtr2 (маска)
gr2	$ m SrcMtr3\ stride$
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

# 7.242 Vec\_Or

## Функции

• void vec Or (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.242.1 Подробное описание

Ядро функции nmppsOr\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N data=}[ar0++gr0] \text{ with data;} \\ \text{rep N data=}[ar1++gr1] \text{ with data or afifo;} \\ \text{rep N } [ar6++gr6] = \text{afifo;} \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

 $7.243 \ \mathrm{Vec}_{\mathrm{Xor}}$  351

# $7.243 \ \mathrm{Vec}_\mathrm{Xor}$

## Функции

• void vec Xor (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.243.1 Подробное описание

Ядро функции nmppsXor\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N data=}[ar0++gr0] \text{ with data;} \\ \text{rep N data=}[ar1++gr1] \text{ with data xor afifo;} \\ \text{rep N }[ar6++gr6]=\text{afifo;} \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	${ m SrcMtr1\ stride}$
ar1	указатель на столбец SrcMtr2
$\operatorname{gr} 1$	${ m SrcMtr2\ stride}$
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для аг6

#### Restrictions:

352 Группы

# 

## Функции

• void vec\_Abs (nmreg nb1, nmreg sb, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.244.1 Подробное описание

Ядро функции nmppsAbs().

Действие функции эквивалентно следующим псевдоинструкциям:

```
 \begin{array}{ll} {\rm rep~N~ram~=[ar0++gr0]} & {\rm with~activate~data;} \\ {\rm rep~N} & {\rm with~vsum~afifo,ram,ram;} \\ {\rm rep~N~[ar6++gr6]=~afifo;} \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

 $7.245~{
m Vec\_Add}$  353

# $7.245 \ \mathrm{Vec}_\mathrm{Add}$

## Функции

• void vec\_Add (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.245.1 Подробное описание

Ядро функции nmppsAdd().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} {\rm rep~N~data}{=}[{\rm ar0}{+}{+}{\rm gr0}]~{\rm with~data};\\ {\rm rep~N~data}{=}[{\rm ar1}{+}{+}{\rm gr1}]~{\rm with~data} + {\rm afifo};\\ {\rm rep~N~[ar6}{+}{+}{\rm gr6}]{=}{\rm afifo}; \end{array}
```

#### 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 = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для аг6

#### Restrictions:

354 Группы

# 7.246 Vec\_ClipExt

## Функции

• void vec\_ClipExt (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg ar2, nmreg ar3, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.246.1 Подробное описание

Ядро функции nmppsClipCC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{lll} \text{rep N ram} &= [ar1]; \\ \text{rep N data} &= [ar0++gr0] \text{ with data-ram}; \\ \text{rep N data} &= [ar2] & \text{with activate afifo+data}; \\ \text{rep N data} &= [ar3] & \text{with activate afifo-data}; \\ \text{rep N } [ar6++gr6] &= \text{afifo}; \\ \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатаель на 64р. слово-константу
ar2	указатаель на 64р. слово-константу
ar3	указатаель на 64р. слово-константу
$\operatorname{gr}5$	Высота матриц $N = [0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для аг6

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

 $nmppsClipCC\_$ 

## 7.247 Vec ClipMul2D2W8 AddVr

#### Функции

• void vec\_ClipMul2D2W8\_AddVr (nmreg nb1, nmreg sb, nmreg f1cr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.247.1 Подробное описание

Взвешенное умножение двух массивов с накоплением и активацей

Ядро функции nmppsClipPowC\_RShift\_Convert\_AddC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\label{eq:continuous_problem} \begin{array}{l} 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; \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 8 строк
f1cr	задает функцию активации
vr	константа для суммирования
ar0	SrcMtr1
gr0	SrcMtr1 stride
ar1	SrcMtr2
$\operatorname{gr} 1$	SrcMtr2 stride
ar4	2 матрицы весовых коэффициентов по 8 64р. слов
$\operatorname{gr} 4$	дублирует nb1
$\operatorname{gr}5$	Высота матриц $N = [0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

# $7.248 \ \mathrm{Vec}\_\mathrm{ClipMulNDNW2}\_\mathrm{AddVr}$

## Функции

• void vec\_ClipMulNDNW2\_AddVr (nmreg nb1, nmreg sb, nmreg f1cr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.248.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацей

Ядро функции SIG Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
 \begin{array}{l} ar2 \! = \! ar0; \\[2mm] gr2 \! = \! [ar1 \! + \! +]; \\[2mm] ar0 \! = \! ar2 \! + \! gr2; \\[2mm] rep \ 2 \ w fifo \! = \! [ar4 \! + \! +], ftw, wtw; \\[2mm] rep \ N \ data = \! [ar0 \! + \! + \! gr0] \ w ith \ v sum \ , activate \ data, vr; \\[2mm] ... \\[
```

#### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 2 строки (sb=2)
f1cr	задает функцию активации
vr	константа для суммирования
ar0	задает базовый адрес для входных массивов (как правило адрес первого массива)
gr0	шаг чтения входного масиива stride for input arrays
ar1	массив адресных смещений входных массивов относительно ar0
gr1	количество массив - К
ar4	массив из К матриц весовых коэффициентов по 2 64р. слов
gr4	дублирует nb1
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

 $\Pi \text{ри выходе из функции изменяется содержимое регистров: } ar 0, ar 1, gr 1, ar 2, gr 2, ar 3, gr 3, ar 4, ar 6, gr 7. \\$ 

## 7.249 Vec ClipMulNDNW4 AddVr

#### Функции

• void vec\_ClipMulNDNW4\_AddVr (nmreg nb1, nmreg sb, nmreg f1cr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.249.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацей

Ядро функции SIG Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
 \begin{array}{l} ar2 \! = \! ar0; \\[2mm] gr2 \! = \! [ar1 \! + \! +]; \\[2mm] ar0 \! = \! ar2 \! + \! gr2; \\[2mm] rep \ 4 \ wfifo \! = \! [ar4 \! + \! +], ftw, wtw; \\[2mm] rep \ N \ data = \! [ar0 \! + \! + \! gr0] \ with \ vsum \ , activate \ data, vr; \\[2mm] ... \\[2mm
```

#### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 4 строки (sb=20002h)
f1cr	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
$\operatorname{gr} 1$	количество массив - К
ar4	массив из К матриц весовых коэффициентов по 4 64р. слов
$\operatorname{gr} 4$	дублирует nb1
$\operatorname{gr}5$	Высота матриц $N=[0,1,231,32,33,]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

#### Restrictions:

 $\Pi$ ри выходе из функции изменяется содержимое регистров: ar0, ar1, gr1, ar2, gr2, ar3, gr3, ar4, ar6, gr7.

## 7.250 Vec ClipMulNDNW8 AddVr

## Функции

• void vec\_ClipMulNDNW8\_AddVr (nmreg nb1, nmreg sb, nmreg f1cr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.250.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацей

Ядро функции SIG Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
 \begin{array}{l} ar2 \! = \! ar0; \\[2mm] gr2 \! = \! [ar1 \! + \! +]; \\[2mm] ar0 \! = \! ar2 \! + \! gr2; \\[2mm] rep \ 8 \ wfifo \! = \! [ar4 \! + \! +], ftw, wtw; \\[2mm] rep \ N \ data = \! [ar0 \! + \! + \! gr0] \ with \ vsum \ , activate \ data, vr; \\[2mm] ... \\[2mm
```

#### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 8 строки (sb=2020202h)
f1cr	задает функцию активации
vr	константа для суммирования
ar0	$\setminus$ задает базовый адрес для входных массивов (как правило адрес первого массива) $\setminus$ set base address for input arrays
gr0	stride for input arrays
ar1	массив адресных смещений входных массивов относительно ar0
gr1	$\setminus$ количество массив - K $\setminus$ number of arrays - K
ar4	массив из К матриц весовых коэффициентов по 8 64р. слов
gr4	дублирует nb1
gr5	Высота матриц $N = [0,1,231,32,33,]$
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.

 $7.251 \text{ Vec\_IncNeg}$  359

# $7.251 \ \mathrm{Vec\_IncNeg}$

## Функции

• void vec\_IncNeg (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.251.1 Подробное описание

Увеличивает отрицательные числа на 1.

Применяется в nmppsDivC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
 \begin{array}{ll} \text{rep N ram} = [\text{ar0} + + \text{gr0}] & \text{with activate data;} \\ \text{rep N} & \text{with ram - afifo;} \\ \text{rep N} & [\text{ar6} + + \text{gr6}] = \text{afifo;} \\ \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	$\mathrm{DstMtr}\ \mathrm{stride}$

#### Restrictions:

# $7.252 \quad \text{Vec\_Mul2D2W1\_AddVr}$

## Функции

• void vec\_Mul2D2W1\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.252.1 Подробное описание

Ядро функции nmppsConvert\_64s(nm64s\* ,nm32s\* ,int ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep 1 wfifo=}[\text{ar4++}], \text{ftw,wtw;} \\ \text{rep N data} = [\text{ar0++gr0}] \text{ with vsum ,data,vr;} \\ \text{rep 1 wfifo=}[\text{ar4++}], \text{ftw,wtw;} \\ \text{rep N data} = [\text{ar1++gr1}] \text{ with vsum ,data,afifo;} \\ \text{rep N } [\text{ar6++gr6}] = \text{afifo;} \\ \end{array}
```

#### 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
$\operatorname{gr} 1$	SrcMtr2 stride
ar4	указатель на 2 матрицы весовых коэффициентов по 2 строки в каждой
gr4	дублирует nb1
gr5	Высота матриц $N=[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

# 7.253 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 gr5, nmreg ar6, nmreg gr6)

### 7.253.1 Подробное описание

Ядро функции nmppsConvert\_32s(nm32s\* ,nm16s\* ,int ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{lll} 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 \ N \ [ar6++gr6] = afifo; \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
1101	задает разонение на колонки (неооходимо wtw)
$^{\mathrm{sb}}$	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 $N=[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

362 Группы

# 7.254 Vec Mul2D2W4 AddVr

## Функции

• void vec\_Mul2D2W4\_AddVr (nmreg nb1, nmreg sb, nmreg f1cr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.254.1 Подробное описание

Ядро функции nmppsConvert\_16s(nm16s\* pSrcVec, nm8s\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep 4 wfifo=}[\text{ar4++}], \text{ftw,wtw;} \\ \text{rep N data} = [\text{ar0++gr0}] \text{ with vsum ,data,vr;} \\ \text{rep 4 wfifo=}[\text{ar4++}], \text{ftw,wtw;} \\ \text{rep N data} = [\text{ar1++gr1}] \text{ with vsum ,data,afifo;} \\ \text{rep N } [\text{ar6++gr6}] = \text{afifo;} \\ \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 4 строки
f1cr	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 $N=[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

# 7.255 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 gr5, nmreg ar6, nmreg gr6)

### 7.255.1 Подробное описание

Применяется в MTR\_Copyua().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{lll} 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 \ N \ [ar6++gr6] = afifo; \end{array}
```

#### 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
$\operatorname{gr} 1$	SrcMtr2 stride
ar4	указатель на 2 матрицы весовых коэффициентов по 8 строк в каждой
gr4	дублирует nb1
gr5	Высота матриц N $= [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

# 7.256 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)

### 7.256.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
\label{eq:continuous_problem} \begin{array}{l} \text{rep 2 wfifo=}[ar4++], ftw, wtw; \\ \text{rep N data} = [ar0++gr0] \text{ with vsum ,} data, vr; \\ \text{rep 2 wfifo=}[ar4++], ftw, wtw; \\ \text{rep N data} = [ar1++gr1] \text{ with vsum ,} data, afifo; \\ \text{rep 2 wfifo=}[ar4++], ftw, wtw; \\ \text{rep N data} = [ar2++gr2] \text{ with vsum ,} data, afifo; \\ \text{rep N } [ar6++gr6] = afifo; \\ \end{array}
```

#### 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 = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.257 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)

### 7.257.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
\label{eq:continuous_section} \begin{array}{l} \operatorname{rep} \ 8 \ \operatorname{wfifo} = [\operatorname{ar4} + +], \operatorname{ftw}, \operatorname{wtw}; \\ \operatorname{rep} \ N \ \operatorname{data} = [\operatorname{ar0} + + \operatorname{gr0}] \ \operatorname{with} \ \operatorname{vsum} \ \operatorname{,data,vr}; \\ \operatorname{rep} \ 8 \ \operatorname{wfifo} = [\operatorname{ar4} + +], \operatorname{ftw}, \operatorname{wtw}; \\ \operatorname{rep} \ N \ \operatorname{data} = [\operatorname{ar1} + + \operatorname{gr1}] \ \operatorname{with} \ \operatorname{vsum} \ \operatorname{,data,afifo}; \\ \operatorname{rep} \ N \ \operatorname{data} = [\operatorname{ar2} + + \operatorname{gr2}] \ \operatorname{with} \ \operatorname{vsum} \ \operatorname{,data,afifo}; \\ \operatorname{rep} \ N \ [\operatorname{ar6} + + \operatorname{gr6}] = \operatorname{afifo}; \\ \end{array}
```

#### 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 $= [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.258 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)

#### 7.258.1 Подробное описание

Ядро функции nmppsConvert 32s(nm32s\* pSrcVec, nm8s\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} {\rm rep~2~wfifo=[ar4++],ftw,wtw;} \\ {\rm rep~N~data=[ar0++gr0]~with~vsum~,data,vr;} \\ {\rm rep~2~wfifo=[ar4++],ftw,wtw;} \\ {\rm rep~N~data=[ar1++gr1]~with~vsum~,data,afifo;} \\ {\rm rep~2~wfifo=[ar4++],ftw,wtw;} \\ {\rm rep~N~data=[ar2++gr2]~with~vsum~,data,afifo;} \\ {\rm rep~2~wfifo=[ar4++],ftw,wtw;} \\ {\rm rep~N~data=[ar3++gr3]~with~vsum~,data,afifo;} \\ {\rm rep~N~data=[ar3++gr3]~with~vsum~,data,afifo;} \\ {\rm rep~N~[ar6++gr6]=afifo;} \end{array}
```

#### 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 $N=[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar3,ar6,gr7.

# $7.259 \ \mathrm{Vec\_MulVN\_AddVN}$

## Функции

• void vec\_MulVN\_AddVN (nmreg nb1, nmreg sb, nmreg f1cr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.259.1 Подробное описание

Ядро функции MTR\_MulC\_AddVsVc().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} {\rm rep~N~data}{=}[{\rm ar0}{+}{+}{\rm gr0}]~{\rm with~vsum~,data,~vr;} \\ {\rm rep~N~data}{=}[{\rm ar1}{+}{+}{\rm gr1}]~{\rm with~afifo}{+}{\rm data;} \\ {\rm rep~N~[ar6}{+}{+}{\rm gr6}]{=}{\rm afifo;} \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
f1cr	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 $N = [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.260 \ \mathrm{Vec\_Sub}$

## Функции

• void vec\_Sub (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.260.1 Подробное описание

Ядро функции nmppsSub().

Действие функции эквивалентно следующим псевдоинструкциям:

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
$\operatorname{gr} 1$	SrcMtr2 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

 $7.261 \text{ Vec\_SubAbs}$  369

# $7.261 \ \mathrm{Vec\_SubAbs}$

## Функции

• void vec\_SubAbs (nmreg nb1, nmreg sb, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.261.1 Подробное описание

Ядро функции nmppsAbsDiff().

Действие функции эквивалентно следующим псевдоинструкциям:

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
f1cr	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 = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.262 \ \mathrm{Vec\_SubVN\_Abs}$

## Функции

• void vec\_SubVN\_Abs (nmreg nb1, nmreg sb, nmreg f1cr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.262.1 Подробное описание

Ядро функции mtr\_SubMV\_Abs().

Действие функции эквивалентно следующим псевдоинструкциям:

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
f1cr	задает функцию активации
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатель на на маску (1 64р. слово)
gr5	Высота матриц $N = [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

7.263 Vec Swap 371

# 7.263 Vec\_Swap

## Функции

• void vec\_Swap (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.263.1 Подробное описание

Ядро функции mtr\_SubVN\_Abs().

Функция осуществляет два одноыременных копирования:

```
\begin{array}{l} [ar0++gr0] => [ar4++gr4] \\ [ar1++gr1] => [ar6++gr6] \\ \text{если ar6} = ar0, gr6 = gr0, \ ar4 = ar1, gr4 = gr1 \\ \text{то выполняется перестановка двух векторов} \\ \mathcal{A}ействие функции эквивалентно следующим псевдоинструкциям:
```

```
\begin{array}{l} \text{rep N ram} = & [ar0++gr0]; \\ \text{rep N data} = & [ar1++gr1] \\ \text{rep N } [ar6++gr6] = & \text{afifo} \\ \text{rep N } [ar4++gr4] = & \text{afifo}; \\ \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	$ m SrcMtr2\ stride$
ar4	указатель на столбец DstMtr1
gr4	DstVec1 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr2
gr6	DstVec2 stride

#### Restrictions:

# 7.264 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)

### 7.264.1 Подробное описание

Ядро функции SIG\_ResizeDown2(nm8u7b\* pSrcVec, nm8u7b\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

#### 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
$\operatorname{gr}4$	дублирует nb1
ar5	указатель на матрицу весовых коэффициентов (16 64р.слов)
gr5	Высота матриц N $= [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar4,ar6,gr2,gr5.

# 7.265 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)

#### 7.265.1 Подробное описание

Ядро функции SIG\_ResizeDown2(nm16u15b\* pSrcVec, nm16u15b\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

#### 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	
$\operatorname{gr} 1$	SrcMtr2 stride	
ar3	указатель на 64р. маску	
$\operatorname{gr}4$	дублирует nb1	
ar5	указатель на матрицу весовых коэффициентов (16 64р.слов)	
gr5	Высота матриц N = $[0,1,231,32,33,]$	
ar6	указатель на столбец DstMtr	
gr6	DstMtr stride	

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar4,ar6,gr2,gr5.

# 7.266 Vec\_not\_data

## Функции

• void vec not data (nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.266.1 Подробное описание

Ядро функции  $nmppsNot_{-}($  ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{lll} \text{rep N data=}[ar0++gr0] \text{ with not data;} \\ \text{rep N } [ar6++gr6]=\text{afifo;} \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

 $7.267 \, \mathrm{Vec\_ram}$  375

# $7.267 \ \mathrm{Vec} \mathrm{\_ram}$

## Функции

• void vec\_ram (nmreg ar0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.267.1 Подробное описание

Ядро функции nmppsSet\_().

Функция служит для заполнения массива константой.

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N ram}{=}[\text{ar0}] \text{ with data;} \\ \text{rep N } [\text{ar6}{+}{+}\text{gr6}]{=}\text{afifo;} \end{array}
```

#### Аргументы

ar0	указатель на столбец SrcMtr1
gr5	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.268 \ \mathrm{Vec\_ram\_sub\_data}$

## Функции

• void vec ram sub data (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.268.1 Подробное описание

Ядро функции nmppsSubCRev().

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} \text{rep N ram=[ar1];} \\ \text{rep N data=[ar0++gr0] with ram-data;} \\ \text{rep N [ar6++gr6]=afifo;} \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
$\operatorname{gr}5$	Высота матриц N = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.269 \quad Vec\_vsum\_activate\_data\_0$

## Функции

• void vec\_vsum\_activate\_data\_0 (nmreg nb1, nmreg sb, nmreg f1cr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.269.1 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
f1cr	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 $N = [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.270 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)

### 7.270.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{ll} rep \ N \ data{=}[ar0{+}{+}gr0] \ with \ vsum \ , data{,}0; \\ rep \ N \ [ar6{+}{+}gr6]{=}afifo; \end{array}
```

#### 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 $N=[0,1,231,32,33,]$	
ar6	указатель на столбец DstMtr	
gr6	DstMtr stride	

#### Restrictions:

# $7.271 \quad Vec\_vsum\_data\_afifo$

## Функции

• void vec\_vsum\_data\_afifo (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr5, nmreg ar6)

### 7.271.1 Подробное описание

Используется в nmppsSum(nm1\* pSrcVec, void\* pTmpBuf, int nSize)

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{tabular}{lll} rep & 1 & data = vfalse; \\ .repeat & N; \\ rep & 1 & data = [ar0 + + gr0] & with vsum , data, afifo; \\ .endrepeat; \\ rep & 1 & [ar6] = afifo; \\ \end{tabular}
```

#### 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 $N = [0,1,231,32,33,]$	
ar6	указатель на столбец DstMtr	

#### Restrictions:

# 7.272 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)

### 7.272.1 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	
f1cr	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 $N=[0,1,231,32,33,]$	
ar6	указатель на столбец DstMtr	
gr6	DstMtr stride	

#### Restrictions:

# $7.273 \quad Vec\_vsum\_shift\_data\_0$

## Функции

• void vec\_vsum\_shift\_data\_0 (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.273.1 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 $N=[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.274 \quad Vec\_vsum\_shift\_data\_vr$

## 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)

### 7.274.1 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 $N = [0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# $7.275 \quad Vec\_vsum\_shift\_data\_afifo$

## Функции

• void vec\_vsum\_shift\_data\_afifo (nmreg nb1, nmreg sb, nmreg f1cr, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

### 7.275.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{lll} rep & 1 & data=vfalse; \\ rep & 1*N & data=[ar0++gr0] & with & vsum \\ , & shift & data,afifo; \\ (rep1 & N & times) \\ rep & 1 & [ar6]=afifo; \\ \end{array}
```

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)	
sb	задает разбиение на 8 строк	
f1cr	задает функцию активации	
woper	в рабочей матрице должны быть загружены весовые коэффициенты	
ar0	указатель на столбец SrcMtr	
gr0	SrcMtr stride	
gr5	Высота матриц $N = [0,1,231,32,33,]$	
ar6	указатель на столбец DstMtr	

#### Restrictions:

# 7.276 Vec\_CompareMinV

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

### Функции

• void vec\_CompareMinV (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr3, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.276.1 Подробное описание

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

```
\label{eq:continuous_problem} $$\operatorname{rep}\ N\ \operatorname{data}=[\operatorname{ar}0++\operatorname{gr}0];$$$ $\operatorname{rep}\ N\ \operatorname{data}=[\operatorname{ar}1++\operatorname{gr}1]$$ with $\operatorname{ram-data};$$$ $\operatorname{rep}\ N\ \operatorname{with\ activate\ afifo};$$$ $\operatorname{rep}\ N\ \operatorname{data}=[\operatorname{ar}3++\operatorname{gr}3]$$ with\ \operatorname{mask\ afifo},\ \operatorname{ram},\ \operatorname{data};$$$$ $\operatorname{rep}\ N\ [\operatorname{ar}6++\operatorname{gr}6]=\operatorname{afifo};$$$$$$$$$$$$$$$
```

задает разбиение на колонки (необходимо wtw)

#### Аргументы

f1cr	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 = $[0,1,231,32,33,]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

#### Restrictions:

# 7.277 Vec CompareMaxV

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

### Функции

• void vec\_CompareMaxV (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr3, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.277.1 Подробное описание

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

```
\label{eq:continuous_problem} $$\operatorname{rep}\ N\ \operatorname{data}=[\operatorname{ar}0++\operatorname{gr}0];$$$ $\operatorname{rep}\ N\ \operatorname{data}=[\operatorname{ar}1++\operatorname{gr}1]$$ with $\operatorname{ram-data};$$$ $\operatorname{rep}\ N\ \operatorname{with\ activate\ afifo};$$$ $\operatorname{rep}\ N\ \operatorname{data}=[\operatorname{ar}3++\operatorname{gr}3]$$ with\ \operatorname{mask\ afifo},\ \operatorname{data},\ \operatorname{ram};$$$$ $\operatorname{rep}\ N\ [\operatorname{ar}6++\operatorname{gr}6]=\operatorname{afifo};$$$$$$$$$$$$$$$
```

задает разбиение на колонки (необходимо wtw)

#### Аргументы

-	1			
f1cr	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 = $[0,1,231,32,33,]$			
ar6	указатель на столбец DstMtr			
gr6	DstMtr stride			

#### Restrictions:

# 7.278 Vec\_DupValueInVector8

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

### Функции

• void vec DupValueInVector8 (nmreg ar1, nmreg gr1)

#### 7.278.1 Подробное описание

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} {\rm gr1} = {\rm gr1} \ {\rm and} \ 0 {\rm FFh}; \\ [{\rm ar1}] = {\rm gr1} + ({\rm gr1} << 8) + ({\rm gr1} << 16) + ({\rm gr1} << 24) + \\ ({\rm gr1} << 32) + ({\rm gr1} << 40) + ({\rm gr1} << 48) + ({\rm gr1} << 56). \\ {\rm ar1} += 2; \end{array}
```

\param ar1

Адрес 64-х битового вектора.

Аргументы

```
gr1 Значение (8 бит).
```

#### Restrictions:

# 7.279 Vec DupValueInVector16

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

### Функции

• void vec DupValueInVector16 (nmreg ar1, nmreg gr1)

#### 7.279.1 Подробное описание

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

```
\begin{array}{l} {\rm gr1} = {\rm gr1} \ {\rm and} \ 0 {\rm FFFFh}; \\ [{\rm ar1}] = {\rm gr1} + ({\rm gr1} << 16) + ({\rm gr1} << 32) + ({\rm gr1} << 48). \\ {\rm ar1} += 2; \end{array}
```

\param ar1

Адрес 64-х битового вектора.

Аргументы

```
gr1 | Значение (8 бит).
```

#### Restrictions:

# $7.280 \quad {\tt Vec\_BuildDiagWeights8}$

Построение диагональной матрицы весовых коеффициентов (8x8).

## Функции

• void vec\_BuildDiagWeights8 (nmreg ar1, nmreg gr1)

## 7.280.1 Подробное описание

Построение диагональной матрицы весовых коеффициентов (8х8).

### Аргументы

ar1	Адрес 64-х буфера весовых коеффициентов (8х64 бит).
$\operatorname{gr} 1$	Значение (8 бит).

#### Restrictions:

# $7.281 \quad \text{Vec\_BuildDiagWeights} 16$

Построение диагональной матрицы весовых коеффициентов (16х16).

## Функции

• void vec\_BuildDiagWeights16 (nmreg ar1, nmreg gr1)

## 7.281.1 Подробное описание

Построение диагональной матрицы весовых коеффициентов (16x16).

#### Аргументы

ar1	Адрес 64-х буфера весовых коеффициентов (4х64 бит).
$\operatorname{gr} 1$	Значение (16 бит).

#### Restrictions:

# $7.282 \quad Vec\_MaxVal\_v8nm8s$

Поиск максимума в 8 байтах

## Функции

• void vec\_MaxVal\_v8nm8s (nmreg ar0, nmreg gr7)

## 7.282.1 Подробное описание

Поиск максимума в 8 байтах

Аргументы

ar0 | Адрес 64р. слова

Возвращаемые значения

gr7 | Максимум из 8 байт

### Restrictions:

 $\Pi$ ри выходе из функции изменяется содержимое регистров:  $\mathrm{gr}0,\,\mathrm{gr}1,\,\mathrm{gr}2,\,\mathrm{gr}3,\,\mathrm{ar}5,\,\mathrm{gr}5,\,\mathrm{gr}7.$ 

# $7.283 \quad \text{Vec\_MaxVal\_v4nm16s}$

Поиск максимума в 4-х 16р. элементах

## Функции

• void vec\_MaxVal\_v4nm16s (nmreg ar0, nmreg gr7)

## 7.283.1 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.

# $7.284~{ m Vec\_MaxVal}$

Поиск максимумов в колонках матрицы SrcMtr1.

## Функции

• void vec\_MaxVal (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr5, nmreg ar6)

## 7.284.1 Подробное описание

Поиск максимумов в колонках матрицы SrcMtr1.

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
f1cr	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,]		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.

# $7.285 \quad \text{Vec\_MinVal\_v8nm8s}$

Поиск минимума в 8 байтах

Функции

• void vec\_MinVal\_v8nm8s (nmreg ar0, nmreg gr7)

## 7.285.1 Подробное описание

Поиск минимума в 8 байтах

Аргументы

ar0 | Адрес 64р. слова |

Возвращаемые значения

gr7 | Максимум из 8 байт

### Restrictions:

 $\Pi$ ри выходе из функции изменяется содержимое регистров:  $\mathrm{gr}0,\,\mathrm{gr}1,\,\mathrm{gr}2,\,\mathrm{gr}3,\,\mathrm{ar}5,\,\mathrm{gr}5,\,\mathrm{gr}7.$ 

# $7.286 \quad Vec\_MinVal\_v4nm16s$

Поиск минимума в 4-х 16р. элементах

## Функции

• void vec\_MinVal\_v4nm16s (nmreg ar0, nmreg gr7)

## 7.286.1 Подробное описание

Поиск минимума в 4-х 16р. элементах

Аргументы

ar0 | Адрес 64р. слова

Возвращаемые значения

gr7 | Максимум из 8 байт

### Restrictions:

 $\Pi$ ри выходе из функции изменяется содержимое регистров:  $\mathrm{gr}0,\,\mathrm{gr}1,\,\mathrm{gr}2,\,\mathrm{gr}3,\,\mathrm{ar}5,\,\mathrm{gr}5,\,\mathrm{gr}7.$ 

 $7.287 \ \mathrm{Vec\_MinVal}$  395

# $7.287 \ \mathrm{Vec}\_\mathrm{MinVal}$

Поиск минимумов в колонках матрицы SrcMtr1.

## Функции

• void vec\_MinVal (nmreg nb1, nmreg f1cr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr5, nmreg ar6)

## 7.287.1 Подробное описание

Поиск минимумов в колонках матрицы SrcMtr1.

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)			
f1cr	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 н	а 64р.	слово	результатов	(максимумов)
-----	-------------	--------	-------	-------------	--------------

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,gr0,ar4,ar3,ar6,gr7.

# 7.288 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)

#### 7.288.1 Подробное описание

Умножение с накоплением

Действие функции эквивалентно следующим псевдоинструкциям:

 $\backslash param\ nb1$ 

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, сосотоящий из 32 длинных слов

#### Аргументы

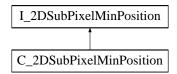
or6	DstMtr	stride
210	DOUNTUL	Bulluc

# Глава 8

# Классы

# 8.1 Класс С 2DSubPixelMinPosition

Граф наследования: C\_2DSubPixelMinPosition:



Открытые члены

- virtual void Find (float \*S9, float &dx, float &dy)
- virtual void Release ()

#### 8.1.1 Подробное описание

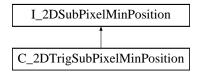
 $\mathrm{Cm}.$ определение в файле isubpixel2dimpl.h строка 7

Объявления и описания членов класса находятся в файле:

• D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h

# 8.2 Класс С 2DTrigSubPixelMinPosition

Граф наследования:  $C_2DTrigSubPixelMinPosition$ :



398

#### 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

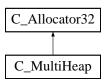
См. определение в файле isubpixel2dimpl.h строка 20

Объявления и описания членов класса находятся в файле:

 $\bullet \ D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h$ 

# 8.3 Класс С\_Allocator32

Граф наследования: С Allocator 32:



#### 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 Fill← Mode=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 $C_{BoxVec} < T >$

- 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

```
\begin{array}{l} template < class \ T > \\ class \ C\_BoxVec < T > \end{array}
```

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 С Неар

```
класс - куча
#include <multiheap.h>
```

- C Heap (void \*addrHeap, size t32 size32Heap)
  - конструктор создает кучу в указанной памяти
- void Create (void \*addrHeap, size\_t32 size32Heap)
  - создает кучу в указанной памяти
- int IsMine (void \*addr)
  - устанавливает принадлежность к куче
- size t32 AllocateMaxAvail ()
  - Возвращает объем свободной памяти в пуле в 32р. словах
- int \* Allocate (size t32 size32Buffer)
  - Выделяет буфер в куче
- int ReleaseBuffer (S\_BufferInfo \*pDelBuffer)
  - удаляет структуру буфера из списка
- int Release (void \*p)
  - осообождат 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 ()
  - Разрещает 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 $C_{Img} < T >$

- 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

```
\begin{array}{l} template < class \ T > \\ class \ C\_Img < \ T > \end{array}
```

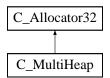
См. определение в файле iSupport.h строка 37

Объявления и описания членов класса находятся в файле:

• D:/GIT/nmpp/include/nmpli/iSupport.h

# 8.8 Класс С\_MultiHeap

Граф наследования: С 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\_t32 size32Heap)

создает кучу по адресууказанного размера (полный размер со служебными данными)

• void \* Allocate (size t32 size32Buffer)

обходит кучи в заднном в AllocateMode порядке и выделяет память заданного размера

• void \* Allocate (size\_t32 size32Buffer, int nPriorHeap0, int nPriorHeap1=-1, int nPriorHeap2=-1, int nPriorHeap3=-1, int nPriorHeap4=-1, int nPriorHeap5=-1)

обходит кучи в задином порядке и выделяет память заданного размера

• void \* AllocateWith (size t32 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 ()

Удваляет все назаблокированные указатели из куч

• void LockHeap (int idxHeap)

Запрещает операции Allocate и Release с кучей

• void UnlockHeap (int idxHeap)

Разрещает операции Allocate и Release с кучей

• int Check ()

#### Открытые атрибуты

• C\_Heap pHeap [MAX\_NUM\_BANKS]

массив куч

• unsigned numHeaps

число проининицализированных 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

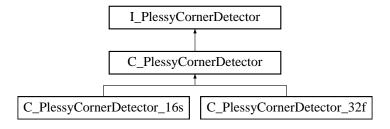
См. определение в файле multiheap.h строка 336

Объявления и описания членов класса находятся в файле:

• D:/GIT/nmpp/include/multiheap.h

# 8.9 Класс C\_PlessyCornerDetector

Граф наследования: С 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 \* Picture 16
- float \* SumSxxSyy
- float \* Subxy
- float S9 [9]
- float threshold
- unsigned char \* cres

#### 8.9.1 Подробное описание

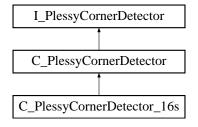
См. определение в файле iPlessyDetector.h строка 8

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

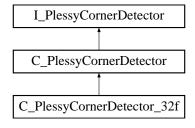
См. определение в файле iPlessyDetector.h строка 56

Объявления и описания членов класса находятся в файле:

 $\bullet \ D:/GIT/nmpp/include/nmpli/iPlessyDetector.h$ 

# 8.11 Kласс 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 \* syy
- float \* Sxx
- float \* Sxv
- 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 С\_RingBufferRemote<br/>< T >

- C\_RingBufferRemote (size\_t ringbuffer\_addr, t\_bytecpy push\_memcopy, t\_bytecpy pop\_  $\hookleftarrow$  memcopy)
- size\_t GetHead ()
- size t GetTail ()
- void SetHead ()
- void SetTail ()
- bool Init (size t ringbuffer addr, t bytecpy push memcopy, t bytecpy pop memcopy)
- \_\_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)
- $\bullet \ \ size\_t \ Pop \ (T \ *pDstElements, \ size\_t \ numElements, \ int \ ExitMode=EXIT\_ON\_COMPLETED)$
- $\bullet \ \ size\_t \ View \ (T*pDstElements, size\_t \ numElements, int \ ExitMode=EXIT\_ON\_COMPLETED)$

#### 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 С\_WarpImg< Т >

- 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

```
\begin{array}{l} template < class \ T > \\ class \ C\_WarpImg < \ T > \end{array}
```

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< nmbits in, nmbits out >

#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 коэффиценты фильтра и инициализирует внутреннюю структуру хранениния коэффициентов в pKernel.

• void Filter (nmbits in \*pSrcImg, nmbits out \*pDstImg, int nImgWidth, int nImgHeight)

Функция одномерной фильтрации

•  $\sim$ CIMG\_FIR ()

Освобождает динамическую область памяти pKernel.

#### Открытые атрибуты

• void(\* pfFree32)(void \*)

Указатель на функции освобождения памяти (pKernel)

• int nKerWidth

Ширина окна коэффицентов КИХ фильтра

• int nKerHeight

Высота окна коэффицентов КИХ фильтра

• nm64s \* pKernel

Указатель на внутреннюю структуру коэффициентов

• int nKernelSize

Размер памяти необходимый для хранения внутренней структуру 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()
```

### Конструктор КИХ фильтра

Выделяет область памяти под внутреннюю структуру коэффицентов

#### 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.
$free 32 \_func$	указатель на функцию динамического особождения памяти

#### 8.14.3 Методы

Функция одномерной фильтрации

$$pDstImg[y][x] = \sum_{i=0}^{nKerHeight-1} \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerHeight/2][x+j-nKerWidth/2] \cdot pWeights[i][j], \\ x = \sum_{j=0}^{nKerHeight-1} \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerHeight/2][x+j-nKerWidth/2] \cdot pWeights[i][j], \\ x = \sum_{j=0}^{nKerHeight-1} \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerHeight/2][x+j-nKerWidth/2] \cdot pWeights[i][j], \\ x = \sum_{j=0}^{nKerHeight-1} \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerHeight/2][x+j-nKerWidth/2] \cdot pWeights[i][j], \\ x = \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerWidth/2][x+j-nKerWidth/2] \cdot pWeights[i][x+j-nKerWidth/2][x+j-nKerWidth/2] \cdot pWeights[i][x+j-nKerWidth/2][x+j-nKerWidth/$$

#### 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 проивзодится выход за границы входного массива pSrcImg. Для коректного поведения функции необходимо дополнительные резервировать поля размером не менее nImgWidth(nKernHeight/2+1) нулевых элементов перед началом и в конце массива pSrcImg.

#### 8.14.3.2 SetWeights()

Загружает коэффиценты фильтра и инициализирует внутреннюю структуру хранениния коэффициентов в pKernel.

#### Аргументы

pWeights	коэффициенты фильтра
nImgWidth	Ширина изображения к которому данный фильтр будет применен. Кратность
	Согласно входному типу.  ———————————————————————————————————

#### Возвращает

указатель на внутреннюю стуктуру 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 пикселов 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

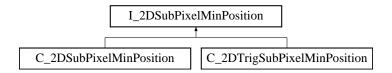
См. определение в файле macros fpu.h строка 21

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

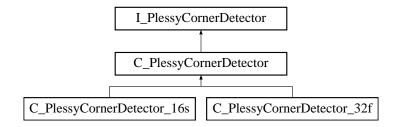
См. определение в файле isubpixel2d.h строка 7

Объявления и описания членов класса находятся в файле:

 $\bullet \ D:/GIT/nmpp/include/nmpli/isubpixel2d.h$ 

## 8.18 Kласс 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^{\land} = (const \ mtr < T > \& matr)
• mtr < T > \& operator^{=} (const T \& val)
• mtr < T > operator^{\wedge} (const mtr < T > \&matr) const
• mtr < T > operator^{\wedge} (const T &val) const
• mtr < T > operator \sim ()
• void set (const T val)
• mtr < T > transpose ()
• mtr < T > & diag (T val)
• T * addr (int y, int x)
• void reset ()
• int sum ()
• \text{vec} < T > \text{get vec} (int y) const
• \text{vec} < T > \text{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)</li>

#### 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

```
\begin{array}{l} template < class \ T > \\ class \ mtr < \ T \ > \end{array}
```

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 nmtype.h строка 1240

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

### 8.24 Kласс 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

```
\begin{array}{l} template < int \ N > \\ class \ nmchar 1D < \ N \ > \end{array}
```

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

```
\begin{array}{l} template < class \ T > \\ class \ nmintpack < \ T > \end{array}
```

См. определение в файле tnmvecpack.h строка 28

Объявления и описания членов класса находятся в файле:

 $\bullet \ D:/GIT/nmpp/include/nmtl/tnmvecpack.h$ 

### 8.28 Шаблон класса nmmtr< T >

```
#include <tnmmtr.h>
```

```
• 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^{\land} = (const \ nmint < T > \&val)
• nmmtr < T > operator^{(n)} (const nmint < T > &val) const
• nmmtr < T > operator \sim () 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 tnmmtr.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

- 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 Ctpyktypa 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

- 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

- 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 Cтруктура 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

- 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

 ${\rm Cm.}$  определение в файле  ${\rm malloc 32.h}$  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<br/>< 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>
```

- 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
- $\bullet$  template<class T2 >
- nmint < T2 > operator\* (const nmvecpack < T2 > &vec) const
- nmvecpack<br/>< 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<br/>< T > operator- (const nmint<br/>< 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 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^{\land} (const nmvecpack < T > &vec) const
   • nmvecpack< T > & operator\sim () 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 >
```

Примеры:

Класс векторов.

```
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

- 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:

 $\bullet \ D{:}/GIT/nmpp/include/nmpli/iDef.h$ 

8.51 Класс RPoint 429

### 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 данамической 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 t32 size 32 Buffer

< размер массива данных в буфере в 32-р словах

• S\_BufferInfo \* pPrevBuffer

< указатель на предыдущий буфер в списке

• S\_BufferInfo \* pNextBuffer

< указатель на следующий буфер в списке

· bool isLocked

< флаг блокировки буфера, запрещаюий его удаление с помощью Release()

• int guardInfoBits1

#### 8.52.1 Подробное описание

класс буфер - заголовок в начале выделяемой данамической 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:

 $\bullet \ D{:}/GIT/nmpp/include/nmtype.h$ 

432

# 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 комплексоного 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 <nmtype.h>

Открытые атрибуты

• unsigned long long vec [8]

#### 8.62.1 Подробное описание

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

См. определение в файле nmtype.h строка 1068

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.63 Структура s\_v16nm32u

#include < nmtype.h >

Открытые атрибуты

• unsigned long long vec [8]

#### 8.63.1 Подробное описание

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

См. определение в файле nmtype.h строка 1205

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.64 Структура s v16nm4u

#include <nmtype.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

436

## 8.67 Ctpyktypa s v2nm32s

#include <nmtype.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 nmtype.h строка 1035

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.68 Структура s v2nm32u

#include < nmtype.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 nmtype.h строка 1172

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.69 Структура s v4nm16s

#include <nmtype.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

438

## 8.72 Структура s v4nm32u

#include <nmtype.h>

Открытые атрибуты

• unsigned long long vec [2]

#### 8.72.1 Подробное описание

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

См. определение в файле nmtype.h строка 1183

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.73 Структура s v4nm8u

#include < nmtype.h >

Открытые атрибуты

• unsigned long long vec [1]

#### 8.73.1 Подробное описание

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

См. определение в файле nmtype.h строка 1106

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.74 Структура s v8nm16s

#include <nmtype.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 <nmtype.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 nmtype.h строка 1194

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.78 Структура s v8nm8s

#include < nmtype.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 nmtype.h строка 977

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.79 Структура s v8nm8u

#include <nmtype.h>

Открытые атрибуты

• unsigned long long vec [1]

#### 8.79.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 8р. 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 Ctpyktypa spot\_struct

Открытые атрибуты

• int Xmin

коородинаты минимального прямоугольника, содержащего пятно

• int Ymin

коородинаты минимального прямоугольника, содержащего пятно

• int Xmax

коородинаты минимального прямоугольника, содержащего пятно

• int Ymax

коородинаты 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 vMin
- · 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, point > 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 indx
- 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 445

## 8.86 Kласс 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 <nmtype.h>

$\cap$		_	
( )TK	рытые	атрибут	ы

• 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 nmtype.h строка 954

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.h

## 8.88 Структура v4nm8s

#include <nmtype.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 nmtype.h строка 966

Объявления и описания членов структуры находятся в файле:

• D:/GIT/nmpp/include/nmtype.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•  $\operatorname{vec} < T > \& \operatorname{operator} += (\operatorname{const} T \& \operatorname{val})$ •  $\operatorname{vec} < T > \& \operatorname{operator} += (\operatorname{const} \operatorname{vec} < T > \& \operatorname{vect})$ vec < T > operator + (const T & val) const• vec < T > operator + (const vec < T > &vect) const• vec < T > & operator = (const T & val) $\operatorname{vec} < T > \& \operatorname{operator} = (\operatorname{const} \operatorname{vec} < T > \& \operatorname{vect})$ vec < T > operator- (const vec < T > &vect) const • vec < T > operator- () const  $\operatorname{vec} < T > \& \operatorname{operator} / = (\operatorname{const} T \operatorname{val})$ • 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  $\operatorname{vec} < T > \& \operatorname{operator} \& = (\operatorname{const} T \& \operatorname{val})$ •  $\operatorname{vec} < T > \& \operatorname{operator} \& = (\operatorname{const} \operatorname{vec} < T > \& \operatorname{vect})$ • vec < T > operator & (const T & val) const • vec < T > operator & (const vec < T > & vect) const•  $\operatorname{vec} < T > \& \operatorname{operator} = (\operatorname{const} T \& \operatorname{val})$ • vec < T > & operator | = (const vec < T > & vect)• vec < T > operator | (const T &val) const • vec < T > operator | (const vec < T > &vect) const•  $\operatorname{vec} < T > \& \operatorname{operator}^{\wedge} = (\operatorname{const} T \& \operatorname{val})$ •  $\operatorname{vec} < T > \& \operatorname{operator}^{\wedge} = (\operatorname{const} \operatorname{vec} < T > \& \operatorname{vect})$ •  $\text{vec} < T > \text{operator}^{\land} \text{ (const } T \text{ \&val) const}$ •  $\text{vec} < T > \text{operator}^{\land} \text{ (const vec} < T > \& \text{vect) const}$ • 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

```
\begin{array}{l} template < class \ T > \\ class \ vec < \ T \ > \end{array}
```

Класс векторов.

Примеры:

```
\begin{array}{l} \inf \ \operatorname{Test}[10] = \{1,125,3,4,5,6,7,8,9,10\}; \\ \inf \ \operatorname{Res} \ [10]; \\ \operatorname{vec} < \operatorname{int} > A0(3); \\ \operatorname{vec} < \operatorname{int} > B0(3); \\ \operatorname{vec} < \operatorname{int} > C0(3); \\ \operatorname{scalar} < \operatorname{int} > a0(2); \\ A0[0] = 1; \\ A0[1] = 2; \\ A0[2] = 3; \\ B0.\operatorname{SetData}(\operatorname{Test}); \\ B0 = A0; \\ a0 = A0[1]; \\ \\ C0 = A0 + B0; \\ C0 = A0 * a0; \\ a0 = A0 * A0; \\ \end{array}
```

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\_dnrm2 (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)

450 Файлы

• 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 A← LPHA, 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  $\bullet \ struct \ s \ v2nm32s \\$ • struct s v4nm32s • struct s v8nm32s• struct  $s_v16nm32s$ • struct  $s_v16nm4u$ • struct s v4nm8u • struct s v8nm8u • struct  $s_v16nm8u$
- struct s\_v16nm16u • struct s\_v2nm32u • struct s\_v4nm32u

struct s\_v4nm16ustruct s\_v8nm16u

- struct s\_v4nm32u
  struct s\_v8nm32u
  struct s\_v16nm32u
  struct nm16sc
- struct s\_nm32sc • struct s\_nm32fc
- struct s\_nm32fcr
- $\bullet \ struct \ s\_nm64sc$
- 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];

452 Файлы

- #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
- $\bullet$  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
- $\bullet$  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 v16nm8sv16nm8s7b
- 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 seq 64
- typedef uint64 fseq64

#### 9.2.1 Подробное описание

Файл, содержащий определения типов упакованных данных.

 $\Phi$ айлы

```
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(_{
m X})
```

Макроопределение:

```
 \begin{array}{c} int \ data[(X)/2]; \\ void \ set(int \ i, \ int \ val) \{ \ ((unsigned \ short*)data)[i] = val; \} \\ short \ operator[] \ (int \ index) \{ \ return \ ((unsigned \ short*)data)[index]; \} \end{array}
```

См. определение в файле nmtype.h строка 933

```
9.2.2.3 VEC_NM16U
```

```
#define VEC_NM16U(X)
```

Макроопределение:

```
 \begin{array}{l} 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]; \} \end{array}
```

См. определение в файле nmtype.h строка 928

```
9.2.2.4 VEC_NM32S
```

```
#define VEC_NM32S(_{
m X})
```

Макроопределение:

```
\begin{array}{c} \mathrm{int} \ \mathrm{data}[(X)]; \ \backslash \\ \ \mathrm{void} \ \mathrm{set}(\mathrm{int} \ i, \ \mathrm{int} \ \mathrm{val}) \{ \ ((\mathrm{int}^*) \mathrm{data})[\mathrm{i}] \! = \! \mathrm{val}; \} \ \backslash \\ \ \mathrm{int} \ \mathrm{operator}[] \ (\mathrm{int} \ \mathrm{index}) \{ \ \mathrm{return} \ ((\mathrm{int}^*) \mathrm{data})[\mathrm{index}]; \} \end{array}
```

См. определение в файле nmtype.h строка 939

```
9.2.2.5 VEC NM4U
```

```
#define VEC_NM4U(X)
```

Макроопределение:

```
 \begin{array}{l} unsigned \; data[(X)/8]; \\ uint4b \; get(int \; nIndex) \{return \; nmget((nm4u^*)data, nIndex); \} \\ uint4b \; operator[] \; (int \; index) \{ \; return \; nmget((nm4u^*)data, index); \} \end{array}
```

См. определение в файле nmtype.h строка 915

```
9.2.2.6 VEC NM8S
```

```
#define VEC_NM8S(X)
```

Макроопределение:

```
 \begin{array}{c} int \; data[(X)/4]; \; \backslash \\ void \; set(int \; i, \; int \; val) \{ \; ((char^*)data)[i] = val; \} \\ char \; operator[] \; (int \; index) \{ \; return \; ((char^*)data)[index]; \} \end{array}
```

См. определение в файле nmtype.h строка 923

456Файлы

# Предметный указатель

Арифметические действия, 114	uint31b, 210
Арифметические операции, 96, 214	$uint32b, \frac{210}{}$
Базовые регистровые функции библиотеки, 325	uint3b, 210
Быстрое преобразование Фурье, 167	$uint4b, \frac{210}{}$
${ m nmppsFFTFree\_32fcr,167}$	$uint63b, \frac{211}{}$
Блочное переупорядочивание, 109	$uint64b, \frac{211}{}$
Целевые функции, 330	uint7b, 211
Элементарные функции, 328	uint8b, 211
Элементарные математические функции, 169	Типы векторных данных, 192
Функции деинтерлейсинга, 97	nm1, 194
IMG DeinterlaceBlend, 97	nm16s, 194
IMG DeinterlaceSplit, 97	nm16s15b, 194
Функции графического вывода текста, 118	nm16u, 195
hex2ascii, 118	nm16u15b, 1 <mark>95</mark>
IMG Print8x15, 119	nm2s, 195
Функции обработки изображений, 322	nm2u, 196
Функции обработки сигналов, 321	nm32s, 196
Функции поддержки, 117, 134, 212	nm32s30b, 196
Инициализация, 92	nm32s31b, 197
Инициализация и копирование, 116, 120, 213	nm32u, 197
Изменение размеров, 166	nm32u31b, 197
КИХ-фильтрация, 99, 181	$\mathrm{nm4s},\mathrm{198}^{'}$
Логические и бинарные операции, 215	nm4u, 198
Масочная фильтрация, 115, 165	nm4u3b, 198
Матричные функции, 320	nm64s, 199
Операции сравнения, 216	nm64s63b, 199
Переупорядочивание и сортировка, 217	nm64u, 199
Переупорядочивание изображений, 108	nm8s, 200
Скалярные функции, 323	nm8s7b, 200
Свертка, 164	nm8u, 200
Типы данных, 318	nm8u7b, 201
Типы скалярных данных, 205	v16nm16s, 201
int15b, 205	v16nm16u, 201
int16b, 206	v16nm32s, 201
int1b, 206	v16nm32u, 201
int2b, 206	v16nm4b3u, $202$
int30b, 206	v16nm4u, 202
,	,
int31b, 207	v16nm8s, 202
int32b, 207	v16nm8s7b, 202
int3b, 207	v16nm8u, 202
int4b, 207	v2nm32s, 202
int63b, 208	v2nm32u, 203
int64b, 208	v4nm16s, 203
int7b, 208	v4nm16u, 203
int8b, 208	v4nm32s, 203
$\operatorname{uint} 15\mathrm{b}, 209$	v4nm32u, $203$
uint 16b, 209	v4nm8u, 203
uint1b, $209$	$v8nm16s, \frac{203}{}$
uint2b, 209	$v8nm16u, \frac{203}{}$

v8nm32s, 204	$ m nmppsFFT2048Fwd\_32fcr,\ 35$
v8nm32u, 204	nmppsFFT2048FwdInitAlloc 32fcr, 35
v8nm8s, 204	FFT-256, 29, 136
v8nm8u, 204	FFT Fwd256, 136
Векторные функции, 319	nmppsFFT256Fwd 32fcr, 29
Векторные функции, 919 Векторно-матричные операции, 135	nmppsFFT256FwdInitAlloc 32fcr, 29
	_ · · · · · · · · · · · · · · · · · · ·
функции взвешенного суммирования, 329	FFT-32, 23
контроль переполнения, 326	${ m nmppsFFT32Fwd\_32fcr,\ 23}$
$\operatorname{GetVec},\ 326$	nmppsFFT32FwdInitAlloc_32fcr, 23
operator <<, 327	FFT-4096, 37, 156
	FFT Fwd4096, 156
${ m Allocate Max Avail}$	$\overline{\text{nmppsFFT4096Fwd}}$ 32fcr, 37
$C\_Heap, 401$	nmppsFFT4096FwdInitAlloc 32fcr, 37
	FFT-512, 31, 141
BLASS-LEVEL1, 66	FFT Fwd512, 141
nm trans, 66	nmppsFFT512Fwd 32fcr, 31
nmblas dasum, 66	
_	nmppsFFT512FwdInitAlloc_32fcr, 31
C 2DSubPixelMinPosition, 397	FFT-64, 25
C 2DTrigSubPixelMinPosition, 397	${ m nmppsFFT64Fwd\_32fcr,\ 25}$
C Allocator 32, 398	nmppsFFT64FwdInitAlloc_32fcr, 25
C  BoxImg < T >, 399	FFT-8192, 160
C = BoxVec < T > 399	FFT_Fwd8192, 160
	FFT-Common, 58
C_Heap, 400	nmppsFFTFwd 32fcr, 58
AllocateMaxAvail, 401	nmppsFFTFwdInitAlloc 32fcr, 58
$C_{\text{Img}} < T >$ , 401	FFT Fwd1024
$C_MultiHeap, 402$	FFT-1024, 146
$C_PlessyCornerDetector, 403$	
C_PlessyCornerDetector_16s, 404	FFT_Fwd2048
C_PlessyCornerDetector_32f, 405	FFT-2048, 151
$C_RingBufferRemote < T >$ , 406	FFT_Fwd256
$C_{\text{WarpImg}} < T >$ , 407	FFT-256, 136
CAPACITY nm64s	${ m FFT\_Fwd4096}$
$nmtype.\overline{h}, 454$	FFT-4096, 156
CIMG FIR< nmbits in, nmbits out >, 408	${ m FFT\_Fwd512}$
CIMG_FIR	FFT-512, 141
	FFT Fwd8192
CIMG_FIR, 409	FFT-8192, 160
Filter, 410	FFT Inv1024
SetWeights, 410	IFFT-1024, 148
D. /CIT / /:l d- / bl b 440	FFT Inv2048
D:/GIT/nmpp/include/nmblas.h, 449	
D:/GIT/nmpp/include/nmtype.h, 451	IFFT-2048, 153
DFT-8, 19	FFT_Inv256
$nmppsDFT8Fwd\_32fcr, 19$	IFFT-256, 138
$ds\_struct, 411$	${ m FFT\_Inv4096}$
	IFFT- $4096, 158$
$\operatorname{EnterHardMode}, 411$	${ m FFT\_Inv}512$
	IFFT-512, 143
FFT-1024, 33, 146	FFT Inv8192
FFT_Fwd1024, 146	
nmpps $FFT1024Fwd 32fcr, 33$	Filter
nmppsFFT1024FwdInitAlloc 32fcr, 33	CIMG FIR, 410
FFT-128, 27	
nmppsFFT128Fwd 32fcr, 27	Fix-point 32, 95
nmppsFFT128FwdInitAlloc_32fcr, 27	Fix-point 64, 94
	FloodFill8
FFT-16, 21	Floodfill, 100
nmppsFFT16Fwd_32fcr, 21	Floodfill, 100
nmppsFFT16FwdInitAlloc_32fcr, 21	FloodFill8, 100
FFT-2048, 35, 151	
$FFT_Fwd2048, 151$	$\operatorname{GetVec}$

контроль переполнения, 326	$s\_nm64sc, 432$
	int15b
hex2ascii	Типы скалярных данных, 205
Функции графического вывода текста, 118	int15in16x4, 412
	int16b
I_2DSubPixelMinPosition, 411	Типы скалярных данных, 206
I_PlessyCornerDetector, 412	int1b
IDFT-8, 39	Типы скалярных данных, 206
nmppsDFT8Inv_32fcr, 39	int2b
IFFT-1024, 52, 148	
FFT Inv1024, 148	Типы скалярных данных, 206
nmppsFFT1024Inv 32fcr, 52	int30b
	Типы скалярных данных, 206
nmppsFFT1024InvInitAlloc_32fcr, 52	int30in32x2, 413
IFFT-128, 46	int31b
nmppsFFT128Inv_32fcr, 46	Типы скалярных данных, 207
${ m nmppsFFT128InvInitAlloc\_32fcr,\ 46}$	int31in32x2, 413
IFFT-16, 40	int32b
nmppsFFT16Inv_32fcr, 40	Типы скалярных данных, 207
nmppsFFT16InvInitAlloc 32fcr, 40	int3b
IFFT-2048, 54, 153	
FFT Inv2048, 153	Типы скалярных данных, 207
nmppsFFT2048Inv 32fcr, 54	int4b
nmppsFFT2048InvInitAlloc 32fcr, 54	Типы скалярных данных, 207
	int63b
IFFT-256, 48, 138	Типы скалярных данных, 208
FFT_Inv256, 138	int64b
${ m nmppsFFT256Inv\_32fcr,\ 48}$	Типы скалярных данных, 208
${ m nmppsFFT256InvInitAlloc\_32fcr,\ 48}$	int7b
IFFT-32, 42	Типы скалярных данных, 208
nmppsFFT32Inv 32fcr, 42	int8b
nmppsFFT32InvInitAlloc 32fcr, 42	Типы скалярных данных, 208
IFFT-4096, 56, 158	<del>-</del>
FFT Inv4096, 158	Integer operations, 93
nmppsFFT4096Inv 32fcr, 56	MTD Add: 121
nmppsFFT4096InvInitAlloc 32fcr, 56	MTR_Addr, 131
IFFT-512, 50, 143	MTR_Copy, 123
	MTR_Copyau, 122
FFT_Inv512, 143	MTR_Free, 130
nmppsFFT512Inv_32fcr, 50	$MTR\_GetVal, 133$
${ m nmppsFFT512InvInitAlloc\_32fcr,\ 50}$	MTR_Malloc, 129
IFFT-64, $44$	MTR_ProdUnitV, 128
${ m nmppsFFT64Inv\_32fcr,\ 44}$	MTR SetVal, 132
nmppsFFT64InvInitAlloc 32fcr, 44	mtr < T >, 414
IFFT-8192, 162	,
FFT_Inv8192, 162	nm1
IFFT-Common, 61	Типы векторных данных, 194
nmppsFFTInv 32fcr, 61	nm16s
nmppsFFTInvInitAlloc_32fcr, 61	Типы векторных данных, 194
	=
IMG_Convert, 106	nm16s15b
IMG_DeinterlaceBlend	Типы векторных данных, 194
Функции деинтерлейсинга, 97	nm16sc, 415
$IMG\_DeinterlaceSplit$	nm16u
$\Phi$ ункции деинтерлейсинга, $97$	Типы векторных данных, 195
IMG Free, 112	nm16u15b
IMG MergeFromBlocks, 111	Типы векторных данных, 195
IMG Print8x15	nm2s
Функции графического вывода текста, 119	Типы векторных данных, 195
IMG RGB32ToGray, 107	nm2u
IMG Release, 113	
	Типы векторных данных, 196
IMG_SplitIntoBlocks, 110	nm32s
im	Типы векторных данных, 196

nm32s30b	nmppcProdC, 69
Типы векторных данных, 196	$\mathrm{nmppcSqrt}, 91$
nm32s31b	m nmppcTblFixArcCos32,~79
Типы векторных данных, 197	nmppcTblFixArcSin32, 78
nm32u	nmppcTblFixCos32, 80
Типы векторных данных, 197	$\mathrm{nmppcTblFixSin}32,81$
nm32u31b	$NmppiFFTSpec\_32fcr, 419$
Типы векторных данных, 197	nmppmCopyua, 121
nm4s	$\operatorname{nmppmMul\_mm},\ 124$
Типы векторных данных, 198	$nmppmMul\_mv\_,126$
nm4u	$nmppmMul\_mv\_\_AddC, 127$
Типы векторных данных, 198	nmppsAbs, 218
nm4u3b	nmppsAbs1, 219
Типы векторных данных, 198	nmppsAbsDiff, 227
nm64s	nmppsAbsDiff1, 228
Типы векторных данных, 199	m nmppsAdd,222
nm64s63b	$\mathrm{nmppsAdd}\_\mathrm{AddC},\ 223$
Типы векторных данных, 199	m nmppsAddC,~221
nm64u	$\mathrm{nmppsAddr}_{\_},306$
Типы векторных данных, 199	nmppsAnd, 244
nm8s	$\mathrm{nmppsAnd4V}_{\_},\ \mathrm{245}$
Типы векторных данных, 200	$\mathrm{nmppsAndNotV}_{\_},\mathrm{246}$
nm8s7b	nmppsAndC, 243
Типы векторных данных, 200	$ m nmppsClipCC\_,  295$
nm8u	$nmppsClipConvert\_AddC\_, 297$
Типы векторных данных, 200	$ m nmppsClipPowC\_, 294$
nm8u7b	nmppsClipRShiftConvert_AddC_, 296
Типы векторных данных, 201	nmppsCmpEq0, 290
$nm\_trans$	$nmppsCmpEq0\_16u15b, \frac{290}{290}$
BLASS-LEVEL1, 66	$nmppsCmpEq0\_32u31b, \frac{291}{2}$
$nmblas\_dasum$	$nmppsCmpEq0\_8u7b, \frac{291}{291}$
BLASS-LEVEL1, 66	${ m nmppsCmpEq0\_16u15b}$
nmchar, 416	nmppsCmpEq0, 290
nmchar1D < N >, 416	${ m nmppsCmpEq0\_32u31b}$
nmchar2D < Y, X >, 417	nmppsCmpEq0, 291
nmintpack $<$ T $>$ , 417	$nmppsCmpEq0\_8u7b$
nmmtr < T >, 418	nmppsCmpEq0, 291
nmppMerge, 315	$ m nmppsCmpEqV\_,~304$
nmppSplit, 314	$nmppsCmpEqC, \frac{299}{}$
nmppSplit_32fcr, 316	nmppsCmpLt0, 289
nmppSplitTmp, 313	$nmppsCmpMinMaxV\_, 292$
nmppcDivC, 68	nmppsCmpNe0, 300
nmppcDoubleToFix32, 73	$nmppsCmpNeV\_, 305$
nmppcDoubleToFix64, 84	nmppsCmpNeC, 301
nmppcFix32ToDouble, 74	nmppsConvert, 263
nmppcFix64Exp01, 89	$nmppsConvert\_1s2s, \frac{264}{}$
nmppcFix64ToDouble, 85	$nmppsConvert\_1u2u, \frac{264}{}$
nmppcFixArcTan32, 72	nmppsConvert_32s32fcr, 264
nmppcFixArcTan64, 88	$nmppsConvert\_32sc32fcr, 265$
nmppcFixDiv64, 86	nmppsConvert_32u32fcr, 265
nmppcFixDivMod32, 82	nmppsConvertRisc_32u8u, 266
nmppcFixExp32, 70	nmppsConvertRisc_8u32u, 266
nmppcFixInv32, 77	nmppsJoin_32f, 266
nmppcFixMul32, 76	nmppsConvert_1s2s
nmppcFixSinCos32, 71	nmppsConvert, 264
nmppcFixSinCos64, 87	nmppsConvert_1u2u
nmppcFixSqrt32, 75	nmppsConvert, 264
nmppcFixSqrt64, 83	m nmppsConvert 32s32fcr

nmppsConvert, 264	IFFT-2048, $54$
$\operatorname{nmppsConvert}\_32\operatorname{sc}32\operatorname{fcr}$	${ m nmppsFFT256Fwd\_32fcr}$
nmppsConvert, 265	FFT-256, 29
nmppsConvert 32u32fcr	${ m nmppsFFT256FwdInitAlloc}$ 32fcr
nmppsConvert, 265	FFT-256, 29
nmppsConvertRisc 32u8u	nmppsFFT256Inv 32fcr
nmppsConvert, 266	IFFT-256, 48
nmppsConvertRisc 8u32u	nmppsFFT256InvInitAlloc 32fcr
nmppsConvert, 266	IFFT-256, 48
nmppsCopy_, 268	nmppsFFT32Fwd 32fcr
nmppsCopyua , 269	FFT-32, 23
nmppsCos 32f, 173	nmppsFFT32FwdInitAlloc 32fcr
nmppsCos 32f, 173	FFT-32, 23
nmppsDFT8Fwd 32fcr	nmppsFFT32Inv 32fcr
DFT-8, 19	IFFT-32, 42
nmppsDFT8Inv 32fcr	nmppsFFT32InvInitAlloc_32fcr
IDFT-8, 39	IFFT-32, 42
nmppsDecimate, 317	nmppsFFT4096Fwd 32fcr
nmppsDisplaceBits, 257	FFT-4096, 37
nmppsDiv 32f, 174	nmppsFFT4096FwdInitAlloc 32fcr
nmppsDiv 32f, 174	FFT-4096, 37
nmppsDivC, 236	nmppsFFT4096Inv 32fcr
	IFFT-4096, 56
nmppsDotProd, 239	
nmppsExp_32f, 175	nmppsFFT4096InvInitAlloc_32fcr
nmppsExp_32f, 175	IFFT-4096, 56
nmppsExp_64f, 176	nmppsFFT512Fwd_32fcr
nmppsExp_64f, 176	FFT-512, 31
nmppsFFT1024Fwd_32fcr	nmppsFFT512FwdInitAlloc_32fcr
FFT-1024, 33	FFT-512, 31
nmppsFFT1024FwdInitAlloc_32fcr	nmppsFFT512Inv_32fcr
FFT-1024, 33	IFFT-512, 50
$nmppsFFT1024Inv\_32fcr$	nmppsFFT512InvInitAlloc_32fcr
IFFT-1024, 52	IFFT-512, 50
$nmppsFFT1024InvInitAlloc\_32fcr$	${ m nmppsFFT64Fwd\_32fcr}$
IFFT-1024, 52	FFT-64, 25
$nmppsFFT128Fwd\_32fcr$	${ m nmppsFFT64FwdInitAlloc\_32fcr}$
FFT-128, 27	FFT-64, 25
$nmppsFFT128FwdInitAlloc\_32fcr$	${ m nmppsFFT64Inv}\_32{ m fcr}$
FFT-128, 27	IFFT-64, 44
$nmppsFFT128Inv\_32fcr$	${ m nmppsFFT64InvInitAlloc\_32fcr}$
IFFT-128, $46$	IFFT-64, 44
$nmppsFFT128InvInitAlloc\_32fcr$	${ m nmppsFFTFree\_32fcr}$
IFFT-128, $46$	Быстрое преобразование Фурье, 167
${\rm nmppsFFT16Fwd\_32fcr}$	${ m nmppsFFTFwd}\_32{ m fcr}$
FFT-16, 21	FFT-Common, 58
$nmppsFFT16FwdInitAlloc\_32fcr$	${ m nmppsFFTFwdInitAlloc\_32fcr}$
FFT-16, 21	FFT-Common, 58
nmppsFFT16Inv 32fcr	${ m nmppsFFTInv\_32fcr}$
IFFT-16, 40	IFFT-Common, 61
nmppsFFT16InvInitAlloc 32fcr	nmppsFFTInvInitAlloc 32fcr
IFFT-16, 40	IFFT-Common, 61
nmppsFFT2048Fwd 32fcr	NmppsFFTSpec, 419
FFT-2048, 35	NmppsFFTSpec 32fcr, 420
nmppsFFT2048FwdInitAlloc 32fcr	nmppsFIR Xs, 182
FFT-2048, 35	nmppsFIRFree, 186
nmppsFFT2048Inv 32fcr	nmppsFIRGetStateSize Xs, 185
IFFT-2048, 54	nmppsFIRInit Xs, 183
nmppsFFT2048InvInitAlloc 32fcr	nmppsFIRInitAlloc Xs, 184
	PP

nmppsFirstNonZeroIndx, 281	m nmppsNeg,220
nmppsFirstZeroIndx, 280	$\mathrm{nmppsNot}\_,\ 242$
NmppsFrame_16s, 420	nmppsOr, 248
NmppsFrame_16u, 420	$\mathrm{nmppsOr3V}_{\_},\mathrm{249}$
NmppsFrame_32s, 421	$\mathrm{nmppsOr4V}_{\_}, 250$
NmppsFrame_32u, 421	m nmppsOrC,~247
NmppsFrame_64s, 421	$nmppsPowx\_64f, 179$
NmppsFrame 64u, 422	$nmppsPowx\_64f, 179$
NmppsFrame 8s, 422	$ m nmppsRShiftC\_, 255$
NmppsFrame 8u, 422	$ m nmppsRShiftC\_AddC\_,~256$
nmppsFree, 65	$nmppsRShiftC\_MulC\_AddC, 233$
nmppsGetVal_, 308	nmppsRShiftC, 254
nmppsGetVal_(return), 309	$\mathrm{nmppsRamp}_{-}, 262$
nmppsJoin 32f	nmppsRand, 90
nmppsConvert, 266	nmppsRandUniform, 260
nmppsLastNonZeroIndx, 283	nmppsRandUniform_64s, 260
nmppsLastZeroIndx, 282	$nmppsRandUniform\_, \frac{261}{}$
nmppsLog 32f, 177	$ m nmppsRandUniform\_64s$
nmppsLog 64f, 177	nmppsRandUniform, 260
nmppsLog 64f, 178	$nmppsRemap\_, 311$
nmppsLog 32f, 177	${ m nmppsSet}$ -инициализация, $259$
nmppsLog 64f, 178	$nmppsSetVal\_, 307$
nmppsMalloc, 64	$nmppsSin\_32f, 172$
NmppsMallocSpec, 423	$nmppsSin\_32f, 172$
nmppsMaskV , 253	nmppsSub, 226
$\frac{1}{1}$ nmppsMax , $\frac{1}{271}$	nmppsSubCRev, 225
nmppsMax 16s15b, 271	nmppsSubC, 224
nmppsMax 32s31b, 272	nmppsSum, 238
nmppsMax 8s7b, 272	nmppsSumN, 235
nmppsMax 16s15b	$\mathrm{nmppsSwap}_{-},270$
nmpsMax , 271	NmppsTmpSpec, 423
nmppsMax 32s31b	nmppsWeightedSum, 241
nmppsMax , 272	$\mathrm{nmppsXor}, 252$
nmppsMax 8s7b	nmppsXorC, 251
nmppsMax , 272	nmreg, 424
nmppsMaxEvery , 285	nmshort, 424
nmppsMaxIndx , 275	nmshort2D < Y, X >, 425
nmppsMin, 273	$\operatorname{nmtype.h}$
nmppsMin 16s15b, 273	$CAPACITY\_nm64s, 454$
nmppsMin 32s31b, 274	$VEC_NM16S, 454$
nmppsMin 8s7b, 274	$VEC_NM16U, 454$
nmppsMin 16s15b	$VEC_NM32S, 454$
nmppsMin, 273	$VEC_NM4U, 455$
nmppsMin 32s31b	$VEC\_NM8S,455$
nmppsMin, 274	nmvecpack < T >, 425
nmppsMin 8s7b	
nmppsMin, 274	operator<<
nmppsMinCmpLtV , 287	контроль переполнения, $327$
nmppsMinEvery_, 284	DCD20 10 407
nmppsMinIndx , 277	RGB32_nm10s, 427
nmppsMinIndxVN , 279	RGB32_nm10u, 427
nmppsMul AddC, 230	RGB32_nm8s, 427
nmppsMulC AddC 2x32s	RGB32_nm8u, 428
nmppsMulC AddC, 231	RGB64_nm16u, 428
nmppsMulC AddV AddC, 234	$ ext{RPoint},429$
	re
nmppsMulC_AddC, 231 nmppsMulC_AddC_2x32s, 231	$s\_nm64sc,\ 432$
	S BufferInfo 400
nmppsMulC, 229	$S_BufferInfo, 429$

S IMG FilterKernel, 430	Типы скалярных данных, 210
S IMG FilterKernel 32s32s, 430	$\mathrm{uint}4\mathrm{b}$
s_int32x2, 431	Типы скалярных данных, 210
s_nm32fc, 431	$\mathrm{uint}63\mathrm{b}$
s_nm32fcr, 431	Типы скалярных данных, 211
$s_nm32sc, 432$	$\mathrm{uint}64\mathrm{b}$
$s_nm64sc, 432$	Типы скалярных данных, 211
im, 432	$\operatorname{uint} 7\mathrm{b}$
re, 432	Типы скалярных данных, 211
s_v16nm16s, 433	$\mathrm{uint}8\mathrm{b}$
s_v16nm16u, 433	Типы скалярных данных, 211
s_v16nm32s, 434	uint8ptr, 445
s_v16nm32u, 434	1616
s_v16nm4u, 434	v16nm16s
s_v16nm8s, 435	Типы векторных данных, 201 v16nm16u
s_v16nm8u, 435	типы векторных данных, 201
s_v2nm32s, 436	v16nm32s
s_v2nm32u, 436	Типы векторных данных, 201
s_v4nm16s, 436	v16nm32u
s_v4nm16u, 437	Типы векторных данных, 201
s_v4nm32s, 437	v16nm4b3u
s_v4nm32u, 438	Типы векторных данных, 202
s_v4nm8u, 438	v16nm4s, 445
s_v8nm16s, 438	v16nm4u
s_v8nm16u, 439	Типы векторных данных, 202
s_v8nm32s, 439	v16nm8s
s_v8nm32u, 440	Типы векторных данных, 202
s_v8nm8s, 440	v16nm8s7b
s_v8nm8u, 440	Типы векторных данных, 202
SIG_CreateResample, 189 SIG Median3, 180	v16nm8u
SIG Resample perf, 191	Типы векторных данных, 202
SIG ResampleDown2, 187	m v2nm32s
SIG ResampleUp3Down2, 188	Типы векторных данных, 202
SIG SetResample, 190	m v2nm32u
SIG XCorr, 170	Типы векторных данных, 203
Set Weights	m v4nm16s
CIMG_FIR, 410	Типы векторных данных, 203
SpecTmp1, 441	${ m v4nm16u}$
spot_struct, 441	Типы векторных данных, 203
• = /	m v4nm32s
tagSegmentInfo, 442	Типы векторных данных, 203
tfixpoint < T, point >, 442	m v4nm32u
Tmp2BuffSpec, 443	Типы векторных данных, 203
	v4nm8s, 446
uint15b	v4nm8u
Типы скалярных данных, 209	Типы векторных данных, 203
uint16b	v8nm16s
Типы скалярных данных, 209	Типы векторных данных, 203
uint16ptr, 444	v8nm16u
uint1b	Типы векторных данных, 203
Типы скалярных данных, 209 uint2b	v8nm32s
	Типы векторных данных, 204 v8nm32u
Типы скалярных данных, 209 uint31b	типы векторных данных, 204
Типы скалярных данных, 210	v8nm8s
иint 32b	Типы векторных данных, 204
Типы скалярных данных, 210	v8nm8u
uint3b	Типы векторных данных, 204
<del></del>	Delite primit gonining 201

VEC_NM16S
$\stackrel{-}{\mathrm{nmtype.h}}$ , 454
VEC_NM16U
nmtype.h, 454
VEC_NM32S nmtype.h, 454
VEC_NM4U
nmtype.h, 455
VEC_NM8S
nmtype.h, 455
$VEC\_QSort, 310$
vec < T >, 446
Vec_0_sub_data, 331
Vec_Abs, 352
Vec_AccMul1D1W32_AddVr, 396 Vec_Add, 353
Vec Add VV shift, 336
Vec And, 348
Vec BuildDiagWeights16, 389
Vec BuildDiagWeights8, 388
Vec ClipExt, 354
Vec_ClipMul2D2W8_AddVr, 355
$Vec\_ClipMulNDNW2\_AddVr, 356$
Vec_ClipMulNDNW4_AddVr, 357
Vec_ClipMulNDNW8_AddVr, 358
Vec_CompareMaxV, 385
Vec_CompareMinV, 384
Vec_DupValueInVector16, 387
Vec_DupValueInVector8, 386
Vec_FilterCoreRow2, 345 Vec_FilterCoreRow4, 346
Vec_FilterCoreRow8, 347
Vec_IncNeg, 359
Vec MUL 2V4toW8 shift, 372
Vec_MUL_2V8toW16_shift, 373
Vec_Mask, 349
Vec_MaxVal, 392
$Vec\_MaxVal\_v4nm16s, 391$
$Vec\_MaxVal\_v8nm8s, 390$
Vec_MinVal, 395
Vec_MinVal_v4nm16s, 394
Vec_MinVal_v8nm8s, 393
Vec_Mul2D2W1_AddVr, 360
Vec_Mul2D2W2_AddVr, 361
Vec_Mul2D2W4_AddVr, 362
Vec_Mul2D2W8_AddVr, 363 Vec_Mul3D3W2_AddVr, 364
Vec_Mul3D3W8_AddVr, 365
Vec_Mul4D4W2_AddVr, 366
Vec_MulVN_AddVN, 367
Vec Or, 350
Vec_Sub, 368
Vec_SubAbs, 369
$Vec\_SubVN\_Abs, 370$
Vec_Swap, 371
Vec_Xor, 351
Vec_activate_data, 332
Vec activate data add 0.333

```
Vec_activate_data_add_ram, 335
Vec_activate_data_xor_data, 334
Vec\_afifo, \frac{337}{}
Vec\_data,\, \color{red}{\bf 338}
Vec_data_add_afifo, 339
Vec data add ram, 340
Vec_data_and_ram, 341
Vec data or ram, 342
Vec_data_sub_ram, 343
Vec_data_xor_ram, 344
Vec\_not\_data,\, \color{red} 374
Vec_ram, 375
Vec_ram_sub_data, 376
Vec_vsum_activate_data_0, 377
Vec_vsum_data_0, 378
Vec_vsum_data_afifo, 379
Vec_vsum_data_vr, 380
Vec_vsum_shift_data_0, 381
Vec\_vsum\_shift\_data\_afifo, 383
Vec_vsum_shift_data_vr, 382
```