

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 | Алфавитный указатель групп | 3 |
| 2.1 | Группы | 3 |
| 3 | Иерархический список классов | 9 |
| 3.1 | Иерархия классов | 9 |
| 4 | Алфавитный указатель классов | 11 |
| 4.1 | Классы | 11 |
| 5 | Список файлов | 13 |
| 5.1 | Файлы | 13 |
| 6 | Группы | 17 |
| 6.1 | DFT-8 | 17 |
| 6.1.1 | Подробное описание | 17 |
| 6.1.2 | Функции | 17 |
| 6.1.2.1 | nmppsDFT8Fwd_32fc() | 17 |
| 6.2 | FFT-16 | 19 |
| 6.2.1 | Подробное описание | 19 |
| 6.2.2 | Функции | 19 |
| 6.2.2.1 | nmppsFFT16Fwd_32fc() | 19 |

| | | |
|---------|---|----|
| 6.2.2.2 | <code>nmppsFFT16FwdInitAlloc_32fc()</code> | 19 |
| 6.3 | FFT-32 | 21 |
| 6.3.1 | Подробное описание | 21 |
| 6.3.2 | Функции | 21 |
| 6.3.2.1 | <code>nmppsFFT32Fwd_32fc()</code> | 21 |
| 6.3.2.2 | <code>nmppsFFT32FwdInitAlloc_32fc()</code> | 21 |
| 6.4 | FFT-64 | 23 |
| 6.4.1 | Подробное описание | 23 |
| 6.4.2 | Функции | 23 |
| 6.4.2.1 | <code>nmppsFFT64Fwd_32fc()</code> | 23 |
| 6.4.2.2 | <code>nmppsFFT64FwdInitAlloc_32fc()</code> | 23 |
| 6.5 | FFT-128 | 25 |
| 6.5.1 | Подробное описание | 25 |
| 6.5.2 | Функции | 25 |
| 6.5.2.1 | <code>nmppsFFT128Fwd_32fc()</code> | 25 |
| 6.5.2.2 | <code>nmppsFFT128FwdInitAlloc_32fc()</code> | 25 |
| 6.6 | FFT-256 | 27 |
| 6.6.1 | Подробное описание | 27 |
| 6.6.2 | Функции | 27 |
| 6.6.2.1 | <code>nmppsFFT256Fwd_32fc()</code> | 27 |
| 6.6.2.2 | <code>nmppsFFT256FwdInitAlloc_32fc()</code> | 27 |
| 6.7 | FFT-512 | 29 |
| 6.7.1 | Подробное описание | 29 |
| 6.7.2 | Функции | 29 |
| 6.7.2.1 | <code>nmppsFFT512Fwd_32fc()</code> | 29 |
| 6.7.2.2 | <code>nmppsFFT512FwdInitAlloc_32fc()</code> | 29 |
| 6.8 | FFT-1024 | 31 |
| 6.8.1 | Подробное описание | 31 |
| 6.8.2 | Функции | 31 |
| 6.8.2.1 | <code>nmppsFFT1024Fwd_32fc()</code> | 31 |

| | | |
|----------|---------------------------------|----|
| 6.8.2.2 | nmppsFFT1024FwdInitAlloc_32fc() | 31 |
| 6.9 | FFT-2048 | 33 |
| 6.9.1 | Подробное описание | 33 |
| 6.9.2 | Функции | 33 |
| 6.9.2.1 | nmppsFFT2048Fwd_32fc() | 33 |
| 6.9.2.2 | nmppsFFT2048FwdInitAlloc_32fc() | 33 |
| 6.10 | IDFT-8 | 35 |
| 6.10.1 | Подробное описание | 35 |
| 6.10.2 | Функции | 35 |
| 6.10.2.1 | nmppsDFT8Inv_32fc() | 35 |
| 6.11 | IFFT-16 | 36 |
| 6.11.1 | Подробное описание | 36 |
| 6.11.2 | Функции | 36 |
| 6.11.2.1 | nmppsFFT16Inv_32fc() | 36 |
| 6.11.2.2 | nmppsFFT16InvInitAlloc_32fc() | 36 |
| 6.12 | IFFT-32 | 38 |
| 6.12.1 | Подробное описание | 38 |
| 6.12.2 | Функции | 38 |
| 6.12.2.1 | nmppsFFT32Inv_32fc() | 38 |
| 6.12.2.2 | nmppsFFT32InvInitAlloc_32fc() | 38 |
| 6.13 | IFFT-64 | 40 |
| 6.13.1 | Подробное описание | 40 |
| 6.13.2 | Функции | 40 |
| 6.13.2.1 | nmppsFFT64Inv_32fc() | 40 |
| 6.13.2.2 | nmppsFFT64InvInitAlloc_32fc() | 40 |
| 6.14 | IFFT-128 | 42 |
| 6.14.1 | Подробное описание | 42 |
| 6.14.2 | Функции | 42 |
| 6.14.2.1 | nmppsFFT128Inv_32fc() | 42 |
| 6.14.2.2 | nmppsFFT128InvInitAlloc_32fc() | 42 |

| | |
|--|----|
| 6.15 IFFT-256 | 44 |
| 6.15.1 Подробное описание | 44 |
| 6.15.2 Функции | 44 |
| 6.15.2.1 nmppsFFT256Inv_32fc() | 44 |
| 6.15.2.2 nmppsFFT256InvInitAlloc_32fc() | 44 |
| 6.16 IFFT-512 | 46 |
| 6.16.1 Подробное описание | 46 |
| 6.16.2 Функции | 46 |
| 6.16.2.1 nmppsFFT512Inv_32fc() | 46 |
| 6.16.2.2 nmppsFFT512InvInitAlloc_32fc() | 46 |
| 6.17 IFFT-1024 | 48 |
| 6.17.1 Подробное описание | 48 |
| 6.17.2 Функции | 48 |
| 6.17.2.1 nmppsFFT1024Inv_32fc() | 48 |
| 6.17.2.2 nmppsFFT1024InvInitAlloc_32fc() | 48 |
| 6.18 IFFT-2048 | 50 |
| 6.18.1 Подробное описание | 50 |
| 6.18.2 Функции | 50 |
| 6.18.2.1 nmppsFFT2048Inv_32fc() | 50 |
| 6.18.2.2 nmppsFFT2048InvInitAlloc_32fc() | 50 |
| 6.19 FFT-Common | 52 |
| 6.19.1 Подробное описание | 52 |
| 6.19.2 Функции | 52 |
| 6.19.2.1 nmppsFFTFwd_32fc() | 52 |
| 6.19.2.2 nmppsFFTFwdInitAlloc_32fc() | 52 |
| 6.20 IFFT-Common | 55 |
| 6.20.1 Подробное описание | 55 |
| 6.20.2 Функции | 55 |
| 6.20.2.1 nmppsFFTInv_32fc() | 55 |
| 6.20.2.2 nmppsFFTInvInitAlloc_32fc() | 55 |

| | | |
|--------|----------------------------------|----|
| 6.21 | <code>nmppsMalloc</code> | 58 |
| 6.21.1 | Подробное описание | 58 |
| 6.22 | <code>nmppsFree</code> | 59 |
| 6.22.1 | Подробное описание | 59 |
| 6.23 | <code>nmppcDivC</code> | 60 |
| 6.23.1 | Подробное описание | 60 |
| 6.24 | <code>nmppcProdC</code> | 61 |
| 6.24.1 | Подробное описание | 61 |
| 6.25 | <code>nmppcFixExp32</code> | 62 |
| 6.25.1 | Подробное описание | 62 |
| 6.26 | <code>nmppcFixSinCos32</code> | 63 |
| 6.26.1 | Подробное описание | 63 |
| 6.27 | <code>nmppcFixArcTan32</code> | 64 |
| 6.27.1 | Подробное описание | 64 |
| 6.28 | <code>nmppcDoubleToFix32</code> | 65 |
| 6.28.1 | Подробное описание | 65 |
| 6.29 | <code>nmppcFix32ToDouble</code> | 66 |
| 6.29.1 | Подробное описание | 66 |
| 6.30 | <code>nmppcFixSqrt32</code> | 67 |
| 6.30.1 | Подробное описание | 67 |
| 6.31 | <code>nmppcFixMul32</code> | 68 |
| 6.31.1 | Подробное описание | 68 |
| 6.32 | <code>nmppcFixInv32</code> | 69 |
| 6.32.1 | Подробное описание | 69 |
| 6.33 | <code>nmppcTblFixArcSin32</code> | 70 |
| 6.33.1 | Подробное описание | 70 |
| 6.34 | <code>nmppcTblFixArcCos32</code> | 71 |
| 6.34.1 | Подробное описание | 71 |
| 6.35 | <code>nmppcTblFixCos32</code> | 72 |
| 6.35.1 | Подробное описание | 72 |

| | | |
|--------|---------------------------------|----|
| 6.36 | <code>nmppcTblFixSin32</code> | 73 |
| 6.36.1 | Подробное описание | 73 |
| 6.37 | <code>nmppcFixDivMod32</code> | 74 |
| 6.37.1 | Подробное описание | 74 |
| 6.38 | <code>nmppcFixSqrt64</code> | 75 |
| 6.38.1 | Подробное описание | 75 |
| 6.39 | <code>nmppcDoubleToFix64</code> | 76 |
| 6.39.1 | Подробное описание | 76 |
| 6.40 | <code>nmppcFix64ToDouble</code> | 77 |
| 6.40.1 | Подробное описание | 77 |
| 6.41 | <code>nmppcFixDiv64</code> | 78 |
| 6.41.1 | Подробное описание | 78 |
| 6.42 | <code>nmppcFixSinCos64</code> | 79 |
| 6.42.1 | Подробное описание | 79 |
| 6.43 | <code>nmppcFixArcTan64</code> | 80 |
| 6.43.1 | Подробное описание | 80 |
| 6.44 | <code>nmppcFix64Exp01</code> | 81 |
| 6.44.1 | Подробное описание | 81 |
| 6.45 | <code>nmppsRand</code> | 82 |
| 6.45.1 | Подробное описание | 82 |
| 6.46 | <code>nmppcSqrt</code> | 83 |
| 6.46.1 | Подробное описание | 83 |
| 6.47 | Инициализация | 84 |
| 6.47.1 | Подробное описание | 84 |
| 6.48 | Integer operations | 85 |
| 6.48.1 | Подробное описание | 85 |
| 6.49 | Fix-point 64 | 86 |
| 6.49.1 | Подробное описание | 86 |
| 6.50 | Fix-point 32 | 87 |
| 6.50.1 | Подробное описание | 87 |

| | | |
|----------|--|-----|
| 6.51 | Арифметические операции | 88 |
| 6.51.1 | Подробное описание | 88 |
| 6.52 | Функции деинтерлейсинга | 89 |
| 6.52.1 | Подробное описание | 89 |
| 6.52.2 | Функции | 89 |
| 6.52.2.1 | IMG_DeinterlaceBlend() | 89 |
| 6.52.2.2 | IMG_DeinterlaceSplit() | 89 |
| 6.53 | КИХ-фильтрация | 91 |
| 6.53.1 | Подробное описание | 91 |
| 6.54 | Floodfill | 92 |
| 6.54.1 | Подробное описание | 92 |
| 6.54.2 | Функции | 92 |
| 6.54.2.1 | FloodFill8() | 93 |
| 6.55 | IMG_Convert | 98 |
| 6.55.1 | Подробное описание | 98 |
| 6.56 | IMG_RGB32ToGray | 99 |
| 6.56.1 | Подробное описание | 99 |
| 6.57 | Переупорядочивание изображений | 100 |
| 6.57.1 | Подробное описание | 100 |
| 6.58 | Блочное переупорядочивание | 101 |
| 6.58.1 | Подробное описание | 101 |
| 6.59 | IMG_SplitIntoBlocks | 102 |
| 6.59.1 | Подробное описание | 102 |
| 6.60 | IMG_MergeFromBlocks | 103 |
| 6.60.1 | Подробное описание | 103 |
| 6.61 | IMG_Free | 104 |
| 6.61.1 | Подробное описание | 104 |
| 6.62 | IMG_Release | 105 |
| 6.62.1 | Подробное описание | 105 |
| 6.63 | Арифметические действия | 106 |

| | |
|---|-----|
| 6.64 Масочная фильтрация | 107 |
| 6.64.1 Подробное описание | 107 |
| 6.65 Инициализация и копирование | 108 |
| 6.65.1 Подробное описание | 108 |
| 6.66 Функции поддержки | 109 |
| 6.66.1 Подробное описание | 109 |
| 6.67 Функции графического вывода текста | 110 |
| 6.67.1 Подробное описание | 110 |
| 6.67.2 Функции | 110 |
| 6.67.2.1 hex2ascii() [1/2] | 110 |
| 6.67.2.2 hex2ascii() [2/2] | 110 |
| 6.67.2.3 IMG_Print8x15() | 111 |
| 6.68 Инициализация и копирование | 112 |
| 6.68.1 Подробное описание | 112 |
| 6.69 nmppmCopyua | 113 |
| 6.69.1 Подробное описание | 113 |
| 6.70 MTR_Copyua | 114 |
| 6.70.1 Подробное описание | 114 |
| 6.71 MTR_Copy | 115 |
| 6.71.1 Подробное описание | 115 |
| 6.72 nmppmMul_mm | 116 |
| 6.72.1 Подробное описание | 116 |
| 6.73 nmppmMul_mv_ | 118 |
| 6.73.1 Подробное описание | 118 |
| 6.74 nmppmMul_mv__AddC | 119 |
| 6.74.1 Подробное описание | 119 |
| 6.75 MTR_ProdUnitV | 120 |
| 6.75.1 Подробное описание | 120 |
| 6.76 MTR_Malloc | 121 |
| 6.77 MTR_Free | 122 |

| | | |
|----------|---------------------------------------|-----|
| 6.77.1 | Подробное описание | 122 |
| 6.78 | MTR_Addr | 123 |
| 6.78.1 | Подробное описание | 123 |
| 6.79 | MTR_SetVal | 124 |
| 6.79.1 | Подробное описание | 124 |
| 6.80 | MTR_GetVal | 125 |
| 6.80.1 | Подробное описание | 125 |
| 6.81 | Функции поддержки | 126 |
| 6.81.1 | Подробное описание | 126 |
| 6.82 | Векторно-матричные операции | 127 |
| 6.82.1 | Подробное описание | 127 |
| 6.83 | FFT-256 | 128 |
| 6.83.1 | Подробное описание | 128 |
| 6.83.2 | Функции | 128 |
| 6.83.2.1 | FFT_Fwd256() | 128 |
| 6.84 | IFFT-256 | 130 |
| 6.84.1 | Подробное описание | 130 |
| 6.84.2 | Функции | 130 |
| 6.84.2.1 | FFT_Inv256() | 130 |
| 6.85 | FFT-512 | 133 |
| 6.85.1 | Подробное описание | 133 |
| 6.85.2 | Функции | 133 |
| 6.85.2.1 | FFT_Fwd512() | 133 |
| 6.86 | IFFT-512 | 135 |
| 6.86.1 | Подробное описание | 135 |
| 6.86.2 | Функции | 135 |
| 6.86.2.1 | FFT_Inv512() | 135 |
| 6.87 | FFT-1024 | 138 |
| 6.87.1 | Подробное описание | 138 |
| 6.87.2 | Функции | 138 |

| | | |
|----------|--------------------|-----|
| 6.87.2.1 | FFT_Fwd1024() | 138 |
| 6.88 | IFFT-1024 | 140 |
| 6.88.1 | Подробное описание | 140 |
| 6.88.2 | Функции | 140 |
| 6.88.2.1 | FFT_Inv1024() | 140 |
| 6.89 | FFT-2048 | 143 |
| 6.89.1 | Подробное описание | 143 |
| 6.89.2 | Функции | 143 |
| 6.89.2.1 | FFT_Fwd2048() | 143 |
| 6.90 | IFFT-2048 | 145 |
| 6.90.1 | Подробное описание | 145 |
| 6.90.2 | Функции | 145 |
| 6.90.2.1 | FFT_Inv2048() | 145 |
| 6.91 | FFT-4096 | 148 |
| 6.91.1 | Подробное описание | 148 |
| 6.91.2 | Функции | 148 |
| 6.91.2.1 | FFT_Fwd4096() | 148 |
| 6.92 | IFFT-4096 | 150 |
| 6.92.1 | Подробное описание | 150 |
| 6.92.2 | Функции | 150 |
| 6.92.2.1 | FFT_Inv4096() | 150 |
| 6.93 | FFT-8192 | 152 |
| 6.93.1 | Подробное описание | 152 |
| 6.93.2 | Функции | 152 |
| 6.93.2.1 | FFT_Fwd8192() | 152 |
| 6.94 | IFFT-8192 | 154 |
| 6.94.1 | Подробное описание | 154 |
| 6.94.2 | Функции | 154 |
| 6.94.2.1 | FFT_Inv8192() | 154 |
| 6.95 | Свертка | 156 |

| | | |
|----------|--|-----|
| 6.95.1 | Подробное описание | 156 |
| 6.96 | Масочная фильтрация | 157 |
| 6.96.1 | Подробное описание | 157 |
| 6.97 | Изменение размеров | 158 |
| 6.97.1 | Подробное описание | 158 |
| 6.98 | Быстрое преобразование Фурье | 159 |
| 6.98.1 | Подробное описание | 159 |
| 6.98.2 | Функции | 159 |
| 6.98.2.1 | nmppsFFTFree_32fc() | 159 |
| 6.99 | SIG_XCorr | 161 |
| 6.99.1 | Подробное описание | 161 |
| 6.100 | SIG_Median3 | 163 |
| 6.100.1 | Подробное описание | 163 |
| 6.101 | КИХ-фильтрация | 164 |
| 6.101.1 | Подробное описание | 164 |
| 6.102 | nmppsFIR_Xs | 165 |
| 6.102.1 | Подробное описание | 165 |
| 6.103 | nmppsFIRInit_Xs | 166 |
| 6.103.1 | Подробное описание | 166 |
| 6.104 | nmppsFIRInitAlloc_Xs | 167 |
| 6.104.1 | Подробное описание | 167 |
| 6.105 | nmppsFIRGetStateSize_Xs | 168 |
| 6.105.1 | Подробное описание | 168 |
| 6.106 | nmppsFIRFree | 169 |
| 6.106.1 | Подробное описание | 169 |
| 6.107 | SIG_ResampleDown2 | 170 |
| 6.107.1 | Подробное описание | 170 |
| 6.108 | SIG_ResampleUp3Down2 | 171 |
| 6.108.1 | Подробное описание | 171 |
| 6.109 | SIG_CreateResample | 172 |

| | |
|--------------------------------------|-----|
| 6.109.1 Подробное описание | 172 |
| 6.110SIG_SetResample | 173 |
| 6.110.1 Подробное описание | 173 |
| 6.111SIG_Resample_perf | 174 |
| 6.111.1 Подробное описание | 174 |
| 6.112Типы векторных данных | 175 |
| 6.112.1 Подробное описание | 176 |
| 6.112.2 Типы | 177 |
| 6.112.2.1 nm1 | 177 |
| 6.112.2.2 nm16s | 177 |
| 6.112.2.3 nm16s15b | 178 |
| 6.112.2.4 nm16u | 178 |
| 6.112.2.5 nm16u15b | 178 |
| 6.112.2.6 nm2s | 179 |
| 6.112.2.7 nm2u | 179 |
| 6.112.2.8 nm32s | 179 |
| 6.112.2.9 nm32s30b | 180 |
| 6.112.2.10nm32s31b | 180 |
| 6.112.2.11nm32u | 180 |
| 6.112.2.12nm32u31b | 181 |
| 6.112.2.13nm4s | 181 |
| 6.112.2.14nm4u | 181 |
| 6.112.2.15nm4u3b | 182 |
| 6.112.2.16nm64s | 182 |
| 6.112.2.17nm64s63b | 182 |
| 6.112.2.18nm64u | 183 |
| 6.112.2.19nm8s | 183 |
| 6.112.2.20nm8s7b | 183 |
| 6.112.2.21nm8u7b | 184 |
| 6.112.2.22nm16s | 184 |

| | |
|---------------------------------------|-----|
| 6.112.2.23 16nm16u | 184 |
| 6.112.2.24 16nm32s | 184 |
| 6.112.2.25 16nm32u | 184 |
| 6.112.2.26 16nm4b3u | 184 |
| 6.112.2.27 16nm4u | 185 |
| 6.112.2.28 16nm8s | 185 |
| 6.112.2.29 16nm8s7b | 185 |
| 6.112.2.30 16nm8u | 185 |
| 6.112.2.31 2nm32s | 185 |
| 6.112.2.32 2nm32u | 185 |
| 6.112.2.33 4nm16s | 185 |
| 6.112.2.34 4nm16u | 186 |
| 6.112.2.35 4nm32s | 186 |
| 6.112.2.36 4nm32u | 186 |
| 6.112.2.37 4nm8u | 186 |
| 6.112.2.38 8nm16s | 186 |
| 6.112.2.39 8nm16u | 186 |
| 6.112.2.40 8nm32s | 186 |
| 6.112.2.41 8nm32u | 186 |
| 6.112.2.42 8nm8s | 186 |
| 6.112.2.43 8nm8u | 186 |
| 6.113 Типы скалярных данных | 187 |
| 6.113.1 Подробное описание | 187 |
| 6.113.2 Типы | 187 |
| 6.113.2.1 int15b | 188 |
| 6.113.2.2 int16b | 188 |
| 6.113.2.3 int1b | 188 |
| 6.113.2.4 int2b | 188 |
| 6.113.2.5 int30b | 189 |
| 6.113.2.6 int31b | 189 |

| | |
|---|-----|
| 6.113.2.7 int32b | 189 |
| 6.113.2.8 int3b | 189 |
| 6.113.2.9 int4b | 190 |
| 6.113.2.10 int63b | 190 |
| 6.113.2.11 int64b | 190 |
| 6.113.2.12 int7b | 190 |
| 6.113.2.13 int8b | 191 |
| 6.113.2.14 int15b | 191 |
| 6.113.2.15 int16b | 191 |
| 6.113.2.16 int1b | 191 |
| 6.113.2.17 int2b | 192 |
| 6.113.2.18 int31b | 192 |
| 6.113.2.19 int32b | 192 |
| 6.113.2.20 int3b | 192 |
| 6.113.2.21 int4b | 193 |
| 6.113.2.22 int63b | 193 |
| 6.113.2.23 int64b | 193 |
| 6.113.2.24 int7b | 193 |
| 6.113.2.25 int8b | 193 |
| 6.114 Функции поддержки | 194 |
| 6.114.1 Подробное описание | 194 |
| 6.115 Инициализация и копирование | 195 |
| 6.115.1 Подробное описание | 195 |
| 6.116 Арифметические операции | 196 |
| 6.116.1 Подробное описание | 196 |
| 6.117 Логические и бинарные операции | 197 |
| 6.117.1 Подробное описание | 197 |
| 6.118 Операции сравнения | 198 |
| 6.118.1 Подробное описание | 198 |
| 6.119 Переупорядочивание и сортировка | 199 |

| | |
|--------------------------------------|-----|
| 6.119.1 Подробное описание | 199 |
| 6.120nmppsAbs | 200 |
| 6.120.1 Подробное описание | 200 |
| 6.121nmppsAbs1 | 201 |
| 6.121.1 Подробное описание | 201 |
| 6.122nmppsNeg | 202 |
| 6.122.1 Подробное описание | 202 |
| 6.123nmppsAddC | 203 |
| 6.123.1 Подробное описание | 203 |
| 6.124nmppsAdd | 204 |
| 6.124.1 Подробное описание | 204 |
| 6.125nmppsAdd_AddC | 205 |
| 6.125.1 Подробное описание | 205 |
| 6.126nmppsSubC | 206 |
| 6.126.1 Подробное описание | 206 |
| 6.127nmppsSubCRev | 207 |
| 6.127.1 Подробное описание | 207 |
| 6.128nmppsSub | 208 |
| 6.128.1 Подробное описание | 208 |
| 6.129nmppsAbsDiff | 209 |
| 6.129.1 Подробное описание | 209 |
| 6.130nmppsAbsDiff1 | 210 |
| 6.130.1 Подробное описание | 210 |
| 6.131nmppsMulC | 211 |
| 6.131.1 Подробное описание | 211 |
| 6.132nmppsMul_AddC | 212 |
| 6.132.1 Подробное описание | 212 |
| 6.133nmppsMulC_AddC | 213 |
| 6.133.1 Подробное описание | 213 |
| 6.133.2 Функции | 213 |

| | |
|----------------------------------|-----|
| 6.133.2.1 nmppsMulC_AddC_2x32s() | 214 |
| 6.134nmppsRShiftC_MulC_AddC | 215 |
| 6.134.1 Подробное описание | 215 |
| 6.135nmppsMulC_AddV_AddC | 216 |
| 6.135.1 Подробное описание | 216 |
| 6.136nmppsSumN | 217 |
| 6.136.1 Подробное описание | 217 |
| 6.137nmppsDivC | 218 |
| 6.137.1 Подробное описание | 218 |
| 6.138nmppsSum | 220 |
| 6.138.1 Подробное описание | 220 |
| 6.139nmppsDotProd | 221 |
| 6.139.1 Подробное описание | 221 |
| 6.140nmppsWeightedSum | 223 |
| 6.140.1 Подробное описание | 223 |
| 6.141nmppsNot_ | 224 |
| 6.141.1 Подробное описание | 224 |
| 6.142nmppsAndC | 225 |
| 6.142.1 Подробное описание | 225 |
| 6.143nmppsAnd | 226 |
| 6.143.1 Подробное описание | 226 |
| 6.144nmppsAnd4V_ | 227 |
| 6.144.1 Подробное описание | 227 |
| 6.145nmppsAndNotV_ | 228 |
| 6.145.1 Подробное описание | 228 |
| 6.146nmppsOrC | 229 |
| 6.146.1 Подробное описание | 229 |
| 6.147nmppsOr | 230 |
| 6.147.1 Подробное описание | 230 |
| 6.148nmppsOr3V_ | 231 |

| | |
|--|-----|
| 6.148.1 Подробное описание | 231 |
| 6.149nmppsOr4V_ | 232 |
| 6.149.1 Подробное описание | 232 |
| 6.150nmppsXorC | 233 |
| 6.150.1 Подробное описание | 233 |
| 6.151nmppsXor | 234 |
| 6.151.1 Подробное описание | 234 |
| 6.152nmppsMaskV_ | 235 |
| 6.152.1 Подробное описание | 235 |
| 6.153nmppsRShiftC | 236 |
| 6.153.1 Подробное описание | 236 |
| 6.154nmppsRShiftC_ | 237 |
| 6.154.1 Подробное описание | 237 |
| 6.155nmppsRShiftC_AddC_ | 238 |
| 6.156nmppsDisplaceBits | 239 |
| 6.156.1 Подробное описание | 239 |
| 6.157nmppsSet-инициализация | 241 |
| 6.157.1 Подробное описание | 241 |
| 6.158nmppsRandUniform | 242 |
| 6.158.1 Подробное описание | 242 |
| 6.158.2 Функции | 242 |
| 6.158.2.1 nmppsRandUniform_64s() | 242 |
| 6.159nmppsRandUniform_ | 243 |
| 6.159.1 Подробное описание | 243 |
| 6.160nmppsRamp_ | 244 |
| 6.160.1 Подробное описание | 244 |
| 6.161nmppsConvert | 245 |
| 6.161.1 Подробное описание | 245 |
| 6.161.2 Функции | 246 |
| 6.161.2.1 nmppsConvert_1s2s() | 246 |

| | |
|------------------------------------|-----|
| 6.161.2.2 nmppsConvert_1u2u() | 246 |
| 6.161.2.3 nmppsConvert_32s32fcr() | 246 |
| 6.161.2.4 nmppsConvert_32sc32fcr() | 247 |
| 6.161.2.5 nmppsConvertRisc_8u32u() | 247 |
| 6.162nmppsCopy_ | 248 |
| 6.162.1 Подробное описание | 248 |
| 6.163nmppsCopyua_ | 249 |
| 6.163.1 Подробное описание | 249 |
| 6.164nmppsSwap_ | 250 |
| 6.164.1 Подробное описание | 250 |
| 6.165nmppsMax_ | 251 |
| 6.165.1 Подробное описание | 251 |
| 6.165.2 Функции | 251 |
| 6.165.2.1 nmppsMax_16s15b() | 252 |
| 6.165.2.2 nmppsMax_32s31b() | 252 |
| 6.165.2.3 nmppsMax_8s7b() | 252 |
| 6.166nmppsMin | 253 |
| 6.166.1 Подробное описание | 253 |
| 6.166.2 Функции | 253 |
| 6.166.2.1 nmppsMin_16s15b() | 254 |
| 6.166.2.2 nmppsMin_32s31b() | 254 |
| 6.166.2.3 nmppsMin_8s7b() | 254 |
| 6.167nmppsMaxIndx_ | 255 |
| 6.167.1 Подробное описание | 255 |
| 6.168nmppsMinIndx_ | 257 |
| 6.168.1 Подробное описание | 257 |
| 6.169nmppsMinIndxVN_ | 259 |
| 6.169.1 Подробное описание | 259 |
| 6.170nmppsFirstZeroIndx | 260 |
| 6.170.1 Подробное описание | 260 |

| | |
|---|-----|
| 6.171nmppsFirstNonZeroIndx | 261 |
| 6.171.1 Подробное описание | 261 |
| 6.172nmppsLastZeroIndx | 262 |
| 6.172.1 Подробное описание | 262 |
| 6.173nmppsLastNonZeroIndx | 263 |
| 6.173.1 Подробное описание | 263 |
| 6.174nmppsMinEvery_ | 264 |
| 6.174.1 Подробное описание | 264 |
| 6.175nmppsMaxEvery_ | 265 |
| 6.175.1 Подробное описание | 265 |
| 6.176nmppsMinCmpLtV_ | 267 |
| 6.176.1 Подробное описание | 267 |
| 6.177nmppsCmpLt0 | 269 |
| 6.177.1 Подробное описание | 269 |
| 6.178nmppsCmpEq0 | 270 |
| 6.178.1 Подробное описание | 270 |
| 6.178.2 Функции | 270 |
| 6.178.2.1 nmppsCmpEq0_16u15b() | 271 |
| 6.178.2.2 nmppsCmpEq0_32u31b() | 271 |
| 6.178.2.3 nmppsCmpEq0_8u7b() | 271 |
| 6.179nmppsCmpMinMaxV_ | 272 |
| 6.179.1 Подробное описание | 272 |
| 6.180nmppsClipPowC_ | 274 |
| 6.180.1 Подробное описание | 274 |
| 6.181nmppsClipCC_ | 275 |
| 6.181.1 Подробное описание | 275 |
| 6.182nmppsClipRShiftConvert_AddC_ | 276 |
| 6.182.1 Подробное описание | 276 |
| 6.183nmppsClipConvert_AddC_ | 277 |
| 6.183.1 Подробное описание | 277 |

| | |
|--------------------------------------|-----|
| 6.184nmppsCmpEqC | 279 |
| 6.184.1 Подробное описание | 279 |
| 6.185nmppsCmpNe0 | 280 |
| 6.185.1 Подробное описание | 280 |
| 6.186nmppsCmpNeC | 281 |
| 6.186.1 Подробное описание | 282 |
| 6.187nmppsCmpEqV_ | 284 |
| 6.187.1 Подробное описание | 284 |
| 6.188nmppsCmpNeV_ | 285 |
| 6.188.1 Подробное описание | 285 |
| 6.189nmppsAddr_ | 286 |
| 6.189.1 Подробное описание | 286 |
| 6.190nmppsSetVal_ | 287 |
| 6.190.1 Подробное описание | 287 |
| 6.191nmppsGetVal_ | 288 |
| 6.191.1 Подробное описание | 288 |
| 6.192nmppsGetVal_ (return) | 289 |
| 6.192.1 Подробное описание | 289 |
| 6.193VEC_ QSort | 290 |
| 6.193.1 Подробное описание | 290 |
| 6.194nmppsRemap_ | 291 |
| 6.194.1 Подробное описание | 291 |
| 6.195nmppSplit | 293 |
| 6.196nmppSplitEco | 294 |
| 6.197Типы данных | 295 |
| 6.197.1 Подробное описание | 295 |
| 6.198Векторные функции | 296 |
| 6.198.1 Подробное описание | 296 |
| 6.199Матричные функции | 297 |
| 6.199.1 Подробное описание | 297 |

| | | |
|-----------|--|-----|
| 6.200 | Функции обработки сигналов | 298 |
| 6.200.1 | Подробное описание | 298 |
| 6.201 | Функции обработки изображений | 299 |
| 6.201.1 | Подробное описание | 299 |
| 6.202 | Скалярные функции | 300 |
| 6.202.1 | Подробное описание | 300 |
| 6.202.1.1 | Введение | 300 |
| 6.203 | Базовые регистровые функции библиотеки | 302 |
| 6.203.1 | Подробное описание | 302 |
| 6.204 | контроль переполнения | 303 |
| 6.204.1 | Подробное описание | 303 |
| 6.204.2 | Макросы | 303 |
| 6.204.2.1 | GetVec | 303 |
| 6.204.3 | Функции | 304 |
| 6.204.3.1 | operator<<() | 304 |
| 6.205 | Элементарные функции | 305 |
| 6.205.1 | Подробное описание | 305 |
| 6.206 | функции взвешенного суммирования | 306 |
| 6.206.1 | Подробное описание | 306 |
| 6.207 | Целевые функции | 307 |
| 6.207.1 | Подробное описание | 307 |
| 6.208 | Vec_0_sub_data | 308 |
| 6.208.1 | Подробное описание | 308 |
| 6.209 | Vec_activate_data | 309 |
| 6.209.1 | Подробное описание | 309 |
| 6.210 | Vec_activate_data_add_0 | 310 |
| 6.210.1 | Подробное описание | 310 |
| 6.211 | Vec_activate_data_xor_data | 311 |
| 6.211.1 | Подробное описание | 311 |
| 6.212 | Vec_activate_data_add_ram | 312 |

| | |
|--------------------------------------|-----|
| 6.212.1 Подробное описание | 312 |
| 6.213Vec_Add_VV_shift | 313 |
| 6.213.1 Подробное описание | 313 |
| 6.214Vec_affo | 314 |
| 6.214.1 Подробное описание | 314 |
| 6.215Vec_data | 315 |
| 6.215.1 Подробное описание | 315 |
| 6.216Vec_data_add_affo | 316 |
| 6.216.1 Подробное описание | 316 |
| 6.217Vec_data_add_ram | 317 |
| 6.217.1 Подробное описание | 317 |
| 6.218Vec_data_and_ram | 318 |
| 6.218.1 Подробное описание | 318 |
| 6.219Vec_data_or_ram | 319 |
| 6.219.1 Подробное описание | 319 |
| 6.220Vec_data_sub_ram | 320 |
| 6.220.1 Подробное описание | 320 |
| 6.221Vec_data_xor_ram | 321 |
| 6.221.1 Подробное описание | 321 |
| 6.222Vec_FilterCoreRow2 | 322 |
| 6.222.1 Подробное описание | 322 |
| 6.223Vec_FilterCoreRow4 | 323 |
| 6.223.1 Подробное описание | 323 |
| 6.224Vec_FilterCoreRow8 | 324 |
| 6.224.1 Подробное описание | 324 |
| 6.225Vec_And | 325 |
| 6.225.1 Подробное описание | 325 |
| 6.226Vec_Mask | 326 |
| 6.226.1 Подробное описание | 326 |
| 6.227Vec_Or | 327 |

| | |
|---------------------------------------|-----|
| 6.227.1 Подробное описание | 327 |
| 6.228Vec_Xor | 328 |
| 6.228.1 Подробное описание | 328 |
| 6.229Vec_Abs | 329 |
| 6.229.1 Подробное описание | 329 |
| 6.230Vec_Add | 330 |
| 6.230.1 Подробное описание | 330 |
| 6.231Vec_ClipExt | 331 |
| 6.231.1 Подробное описание | 331 |
| 6.232Vec_ClipMul2D2W8_AddVr | 332 |
| 6.232.1 Подробное описание | 332 |
| 6.233Vec_ClipMulNDNW2_AddVr | 333 |
| 6.233.1 Подробное описание | 333 |
| 6.234Vec_ClipMulNDNW4_AddVr | 334 |
| 6.234.1 Подробное описание | 334 |
| 6.235Vec_ClipMulNDNW8_AddVr | 335 |
| 6.235.1 Подробное описание | 335 |
| 6.236Vec_IncNeg | 336 |
| 6.236.1 Подробное описание | 336 |
| 6.237Vec_Mul2D2W1_AddVr | 337 |
| 6.237.1 Подробное описание | 337 |
| 6.238Vec_Mul2D2W2_AddVr | 338 |
| 6.238.1 Подробное описание | 338 |
| 6.239Vec_Mul2D2W4_AddVr | 339 |
| 6.239.1 Подробное описание | 339 |
| 6.240Vec_Mul2D2W8_AddVr | 340 |
| 6.240.1 Подробное описание | 340 |
| 6.241Vec_Mul3D3W2_AddVr | 341 |
| 6.241.1 Подробное описание | 341 |
| 6.242Vec_Mul3D3W8_AddVr | 342 |

| | |
|---|-----|
| 6.242.1 Подробное описание | 342 |
| 6.243Vec_Mul4D4W2_AddVr | 343 |
| 6.243.1 Подробное описание | 343 |
| 6.244Vec_MulVN_AddVN | 344 |
| 6.244.1 Подробное описание | 344 |
| 6.245Vec_Sub | 345 |
| 6.245.1 Подробное описание | 345 |
| 6.246Vec_SubAbs | 346 |
| 6.246.1 Подробное описание | 346 |
| 6.247Vec_SubVN_Abs | 347 |
| 6.247.1 Подробное описание | 347 |
| 6.248Vec_Swap | 348 |
| 6.248.1 Подробное описание | 348 |
| 6.249Vec_MUL_2V4toW8_shift | 349 |
| 6.249.1 Подробное описание | 349 |
| 6.250Vec_MUL_2V8toW16_shift | 350 |
| 6.250.1 Подробное описание | 350 |
| 6.251Vec_not_data | 351 |
| 6.251.1 Подробное описание | 351 |
| 6.252Vec_ram | 352 |
| 6.252.1 Подробное описание | 352 |
| 6.253Vec_ram_sub_data | 353 |
| 6.253.1 Подробное описание | 353 |
| 6.254Vec_vsum_activate_data_0 | 354 |
| 6.254.1 Подробное описание | 354 |
| 6.255Vec_vsum_data_0 | 355 |
| 6.255.1 Подробное описание | 355 |
| 6.256Vec_vsum_data_affo | 356 |
| 6.256.1 Подробное описание | 356 |
| 6.257Vec_vsum_data_vr | 357 |

| | |
|--|-----|
| 6.257.1 Подробное описание | 357 |
| 6.258Vec_vsum_shift_data_0 | 358 |
| 6.258.1 Подробное описание | 358 |
| 6.259Vec_vsum_shift_data_vr | 359 |
| 6.259.1 Подробное описание | 359 |
| 6.260Vec_vsum_shift_data_afifo | 360 |
| 6.260.1 Подробное описание | 360 |
| 6.261Vec_CompareMinV | 361 |
| 6.261.1 Подробное описание | 361 |
| 6.262Vec_CompareMaxV | 362 |
| 6.262.1 Подробное описание | 362 |
| 6.263Vec_DupValueInVector8 | 363 |
| 6.263.1 Подробное описание | 363 |
| 6.264Vec_DupValueInVector16 | 364 |
| 6.264.1 Подробное описание | 364 |
| 6.265Vec_BuildDiagWeights8 | 365 |
| 6.265.1 Подробное описание | 365 |
| 6.266Vec_BuildDiagWeights16 | 366 |
| 6.266.1 Подробное описание | 366 |
| 6.267Vec_MaxVal_v8nm8s | 367 |
| 6.267.1 Подробное описание | 367 |
| 6.268Vec_MaxVal_v4nm16s | 368 |
| 6.268.1 Подробное описание | 368 |
| 6.269Vec_MaxVal | 369 |
| 6.269.1 Подробное описание | 369 |
| 6.270Vec_MinVal_v8nm8s | 370 |
| 6.270.1 Подробное описание | 370 |
| 6.271Vec_MinVal_v4nm16s | 371 |
| 6.271.1 Подробное описание | 371 |
| 6.272Vec_MinVal | 372 |
| 6.272.1 Подробное описание | 372 |
| 6.273Vec_AccMul1D1W32_AddVr | 373 |
| 6.273.1 Подробное описание | 373 |

| | | |
|----------|--|-----|
| 7 | Классы | 375 |
| 7.1 | Класс <code>C_2DSubPixelMinPosition</code> | 375 |
| 7.1.1 | Подробное описание | 375 |
| 7.2 | Класс <code>C_2DTrigSubPixelMinPosition</code> | 375 |
| 7.2.1 | Подробное описание | 376 |
| 7.3 | Класс <code>C_Allocator32</code> | 376 |
| 7.3.1 | Подробное описание | 376 |
| 7.4 | Шаблон класса <code>C_BoxImg< T ></code> | 377 |
| 7.4.1 | Подробное описание | 377 |
| 7.5 | Шаблон класса <code>C_BoxVec< T ></code> | 377 |
| 7.5.1 | Подробное описание | 378 |
| 7.6 | Класс <code>C_Heap</code> | 378 |
| 7.6.1 | Подробное описание | 379 |
| 7.6.2 | Методы | 379 |
| 7.6.2.1 | <code>AllocateMaxAvail()</code> | 379 |
| 7.7 | Шаблон класса <code>C_Img< T ></code> | 379 |
| 7.7.1 | Подробное описание | 380 |
| 7.8 | Класс <code>C_MultiHeap</code> | 380 |
| 7.8.1 | Подробное описание | 381 |
| 7.9 | Класс <code>C_PlessyCornerDetector</code> | 381 |
| 7.9.1 | Подробное описание | 382 |
| 7.10 | Класс <code>C_PlessyCornerDetector_16s</code> | 382 |
| 7.10.1 | Подробное описание | 383 |
| 7.11 | Класс <code>C_PlessyCornerDetector_32f</code> | 383 |
| 7.11.1 | Подробное описание | 384 |
| 7.12 | Шаблон класса <code>C_RingBuffer< T ></code> | 384 |
| 7.12.1 | Подробное описание | 385 |
| 7.12.2 | Методы | 385 |
| 7.12.2.1 | <code>PushRequest()</code> | 385 |
| 7.13 | Шаблон класса <code>C_RingBufferRemote< T ></code> | 386 |

| | | |
|----------|--|-----|
| 7.13.1 | Подробное описание | 387 |
| 7.14 | Шаблон класса <code>C_WarpImg< T ></code> | 387 |
| 7.14.1 | Подробное описание | 387 |
| 7.15 | Шаблон класса <code>CIMG_FIR< nmbits_in, nmbits_out ></code> | 387 |
| 7.15.1 | Подробное описание | 388 |
| 7.15.2 | Конструктор(ы) | 388 |
| 7.15.2.1 | <code>CIMG_FIR()</code> | 389 |
| 7.15.3 | Методы | 389 |
| 7.15.3.1 | <code>Filter()</code> | 389 |
| 7.15.3.2 | <code>SetWeights()</code> | 390 |
| 7.16 | Структура <code>ds_struct</code> | 390 |
| 7.16.1 | Подробное описание | 390 |
| 7.17 | Класс <code>I_2DSubPixelMinPosition</code> | 391 |
| 7.17.1 | Подробное описание | 391 |
| 7.18 | Класс <code>I_PlessyCornerDetector</code> | 391 |
| 7.18.1 | Подробное описание | 392 |
| 7.19 | Структура <code>int15in16x4</code> | 392 |
| 7.19.1 | Подробное описание | 392 |
| 7.20 | Структура <code>int30in32x2</code> | 392 |
| 7.20.1 | Подробное описание | 392 |
| 7.21 | Структура <code>int31in32x2</code> | 392 |
| 7.21.1 | Подробное описание | 393 |
| 7.22 | Шаблон класса <code>mtr< T ></code> | 393 |
| 7.22.1 | Подробное описание | 394 |
| 7.23 | Структура <code>nm16sc</code> | 394 |
| 7.23.1 | Подробное описание | 395 |
| 7.24 | Класс <code>nmchar</code> | 395 |
| 7.24.1 | Подробное описание | 395 |
| 7.25 | Шаблон класса <code>nmchar1D< N ></code> | 395 |
| 7.25.1 | Подробное описание | 396 |

| | | |
|--------|---|-----|
| 7.26 | Шаблон класса <code>nmchar2D< Y, X ></code> | 396 |
| 7.26.1 | Подробное описание | 396 |
| 7.27 | Шаблон класса <code>nmintpack< T ></code> | 396 |
| 7.27.1 | Подробное описание | 397 |
| 7.28 | Шаблон класса <code>nmtr< T ></code> | 397 |
| 7.28.1 | Подробное описание | 398 |
| 7.29 | Шаблон класса <code>nmtrpack< T ></code> | 398 |
| 7.29.1 | Подробное описание | 400 |
| 7.30 | Структура <code>NmpriFFTSpec_32fc</code> | 400 |
| 7.30.1 | Подробное описание | 400 |
| 7.31 | Структура <code>NmprsFFTSpec</code> | 400 |
| 7.31.1 | Подробное описание | 401 |
| 7.32 | Структура <code>NmprsFFTSpec_32fc</code> | 401 |
| 7.32.1 | Подробное описание | 401 |
| 7.33 | Структура <code>NmprsFrame_16s</code> | 401 |
| 7.33.1 | Подробное описание | 401 |
| 7.34 | Структура <code>NmprsFrame_16u</code> | 401 |
| 7.34.1 | Подробное описание | 402 |
| 7.35 | Структура <code>NmprsFrame_32s</code> | 402 |
| 7.35.1 | Подробное описание | 402 |
| 7.36 | Структура <code>NmprsFrame_32u</code> | 402 |
| 7.36.1 | Подробное описание | 402 |
| 7.37 | Структура <code>NmprsFrame_64s</code> | 402 |
| 7.37.1 | Подробное описание | 403 |
| 7.38 | Структура <code>NmprsFrame_64u</code> | 403 |
| 7.38.1 | Подробное описание | 403 |
| 7.39 | Структура <code>NmprsFrame_8s</code> | 403 |
| 7.39.1 | Подробное описание | 403 |
| 7.40 | Структура <code>NmprsFrame_8u</code> | 403 |
| 7.40.1 | Подробное описание | 404 |

| | |
|--|-----|
| 7.41 Структура NmprrsMallocSpec | 404 |
| 7.41.1 Подробное описание | 404 |
| 7.42 Структура NmprrsTmpSpec | 404 |
| 7.42.1 Подробное описание | 404 |
| 7.43 Структура nmreg | 405 |
| 7.43.1 Подробное описание | 405 |
| 7.44 Класс nmshort | 405 |
| 7.44.1 Подробное описание | 405 |
| 7.45 Шаблон класса nmshort2D< Y, X > | 406 |
| 7.45.1 Подробное описание | 406 |
| 7.46 Шаблон класса nmvespack< T > | 406 |
| 7.46.1 Подробное описание | 407 |
| 7.47 Структура RGB24_nm8u | 408 |
| 7.47.1 Подробное описание | 408 |
| 7.48 Структура RGB32_nm10s | 408 |
| 7.48.1 Подробное описание | 408 |
| 7.49 Структура RGB32_nm10u | 408 |
| 7.49.1 Подробное описание | 409 |
| 7.50 Структура RGB32_nm8s | 409 |
| 7.50.1 Подробное описание | 409 |
| 7.51 Структура RGB32_nm8u | 409 |
| 7.51.1 Подробное описание | 409 |
| 7.52 Структура RGB64_nm16u | 410 |
| 7.52.1 Подробное описание | 410 |
| 7.53 Класс RPoint | 410 |
| 7.53.1 Подробное описание | 410 |
| 7.54 Структура S_BufferInfo | 410 |
| 7.54.1 Подробное описание | 411 |
| 7.55 Структура S_IMG_FilterKernel | 411 |
| 7.55.1 Подробное описание | 411 |

| | | |
|----------|--|-----|
| 7.56 | Структура <code>S_IMG_FilterKernel_32s32s</code> | 412 |
| 7.56.1 | Подробное описание | 412 |
| 7.57 | Структура <code>s_int32x2</code> | 412 |
| 7.57.1 | Подробное описание | 412 |
| 7.58 | Структура <code>s_nm32fc</code> | 412 |
| 7.58.1 | Подробное описание | 412 |
| 7.59 | Структура <code>s_nm32fcr</code> | 413 |
| 7.59.1 | Подробное описание | 413 |
| 7.60 | Структура <code>s_nm32sc</code> | 413 |
| 7.60.1 | Подробное описание | 413 |
| 7.61 | Структура <code>s_nm64sc</code> | 413 |
| 7.61.1 | Подробное описание | 413 |
| 7.61.2 | Данные класса | 413 |
| 7.61.2.1 | <code>im</code> | 414 |
| 7.61.2.2 | <code>re</code> | 414 |
| 7.62 | Структура <code>s_v16nm16s</code> | 414 |
| 7.62.1 | Подробное описание | 414 |
| 7.63 | Структура <code>s_v16nm16u</code> | 414 |
| 7.63.1 | Подробное описание | 415 |
| 7.64 | Структура <code>s_v16nm32s</code> | 415 |
| 7.64.1 | Подробное описание | 415 |
| 7.65 | Структура <code>s_v16nm32u</code> | 415 |
| 7.65.1 | Подробное описание | 416 |
| 7.66 | Структура <code>s_v16nm4u</code> | 416 |
| 7.66.1 | Подробное описание | 416 |
| 7.67 | Структура <code>s_v16nm8s</code> | 416 |
| 7.67.1 | Подробное описание | 416 |
| 7.68 | Структура <code>s_v16nm8u</code> | 417 |
| 7.68.1 | Подробное описание | 417 |
| 7.69 | Структура <code>s_v2nm32s</code> | 417 |

| | | |
|--------|---|-----|
| 7.69.1 | Подробное описание | 417 |
| 7.70 | Структура <code>s_v2nm32u</code> | 417 |
| 7.70.1 | Подробное описание | 418 |
| 7.71 | Структура <code>s_v4nm16s</code> | 418 |
| 7.71.1 | Подробное описание | 418 |
| 7.72 | Структура <code>s_v4nm16u</code> | 418 |
| 7.72.1 | Подробное описание | 418 |
| 7.73 | Структура <code>s_v4nm32s</code> | 419 |
| 7.73.1 | Подробное описание | 419 |
| 7.74 | Структура <code>s_v4nm32u</code> | 419 |
| 7.74.1 | Подробное описание | 419 |
| 7.75 | Структура <code>s_v4nm8u</code> | 419 |
| 7.75.1 | Подробное описание | 420 |
| 7.76 | Структура <code>s_v8nm16s</code> | 420 |
| 7.76.1 | Подробное описание | 420 |
| 7.77 | Структура <code>s_v8nm16u</code> | 420 |
| 7.77.1 | Подробное описание | 420 |
| 7.78 | Структура <code>s_v8nm32s</code> | 421 |
| 7.78.1 | Подробное описание | 421 |
| 7.79 | Структура <code>s_v8nm32u</code> | 421 |
| 7.79.1 | Подробное описание | 421 |
| 7.80 | Структура <code>s_v8nm8s</code> | 421 |
| 7.80.1 | Подробное описание | 422 |
| 7.81 | Структура <code>s_v8nm8u</code> | 422 |
| 7.81.1 | Подробное описание | 422 |
| 7.82 | Структура <code>SpecTmp1</code> | 422 |
| 7.82.1 | Подробное описание | 422 |
| 7.83 | Структура <code>spot_struct</code> | 423 |
| 7.83.1 | Подробное описание | 423 |
| 7.84 | Структура <code>tagSegmentInfo</code> | 423 |

| | | |
|---------|--|-----|
| 7.84.1 | Подробное описание | 423 |
| 7.85 | Шаблон класса <code>tfixpoint< T, point ></code> | 424 |
| 7.85.1 | Подробное описание | 424 |
| 7.86 | Структура <code>Tmp2BuffSpec</code> | 425 |
| 7.86.1 | Подробное описание | 425 |
| 7.87 | Класс <code>uint16ptr</code> | 425 |
| 7.87.1 | Подробное описание | 426 |
| 7.88 | Класс <code>uint8ptr</code> | 426 |
| 7.88.1 | Подробное описание | 426 |
| 7.89 | Структура <code>v16nm4s</code> | 427 |
| 7.89.1 | Подробное описание | 427 |
| 7.90 | Структура <code>v4nm8s</code> | 427 |
| 7.90.1 | Подробное описание | 427 |
| 7.91 | Шаблон класса <code>vec< T ></code> | 427 |
| 7.91.1 | Подробное описание | 429 |
| 8 | Файлы | 431 |
| 8.1 | Файл <code>D:/GIT/nmpp/include/nmtype.h</code> | 431 |
| 8.1.1 | Подробное описание | 434 |
| 8.1.2 | Макросы | 434 |
| 8.1.2.1 | <code>CAPACITY_nm64s</code> | 434 |
| 8.1.2.2 | <code>VEC_NM16S</code> | 434 |
| 8.1.2.3 | <code>VEC_NM16U</code> | 434 |
| 8.1.2.4 | <code>VEC_NM32S</code> | 435 |
| 8.1.2.5 | <code>VEC_NM4U</code> | 435 |
| 8.1.2.6 | <code>VEC_NM8S</code> | 435 |
| | Алфавитный указатель | 437 |

Глава 1

Введение

1.1 Introduction

This is the introduction.

1.2 Installation

1.2.1 Step 1: Opening the box

etc...

Глава 2

Алфавитный указатель групп

2.1 Группы

Полный список групп.

| | |
|---------------------------------------|-----|
| FFT-256 | 128 |
| IFFT-256 | 130 |
| FFT-512 | 133 |
| IFFT-512 | 135 |
| FFT-1024 | 138 |
| IFFT-1024 | 140 |
| FFT-2048 | 143 |
| IFFT-2048 | 145 |
| FFT-4096 | 148 |
| IFFT-4096 | 150 |
| FFT-8192 | 152 |
| IFFT-8192 | 154 |
| Типы данных | 295 |
| Типы векторных данных | 175 |
| Типы скалярных данных | 187 |
| Векторные функции | 296 |
| Функции поддержки | 194 |
| nmppsMalloc | 58 |
| nmppsFree | 59 |
| nmppsAddr_ | 286 |
| nmppsSetVal_ | 287 |
| nmppsGetVal_ | 288 |
| nmppsGetVal_(return) | 289 |
| Инициализация и копирование | 195 |
| nmppsSet-инициализация | 241 |
| nmppsRandUniform | 242 |
| nmppsRandUniform_ | 243 |
| nmppsRamp_ | 244 |
| nmppsConvert | 245 |
| nmppsCopy_ | 248 |
| nmppsCopyua_ | 249 |
| nmppsSwap_ | 250 |
| Арифметические операции | 196 |
| nmppsAbs | 200 |
| nmppsAbs1 | 201 |

| | |
|--|-----|
| nmppsNeg | 202 |
| nmppsAddC | 203 |
| nmppsAdd | 204 |
| nmppsAdd_AddC | 205 |
| nmppsSubC | 206 |
| nmppsSubCRev | 207 |
| nmppsSub | 208 |
| nmppsAbsDiff | 209 |
| nmppsAbsDiff1 | 210 |
| nmppsMulC | 211 |
| nmppsMul_AddC | 212 |
| nmppsMulC_AddC | 213 |
| nmppsRShiftC_MulC_AddC | 215 |
| nmppsMulC_AddV_AddC | 216 |
| nmppsSumN | 217 |
| nmppsDivC | 218 |
| nmppsSum | 220 |
| nmppsDotProd | 221 |
| nmppsWeightedSum | 223 |
| Логические и бинарные операции | 197 |
| nmppsNot_ | 224 |
| nmppsAndC | 225 |
| nmppsAnd | 226 |
| nmppsAnd4V_ | 227 |
| nmppsAndNotV_ | 228 |
| nmppsOrC | 229 |
| nmppsOr | 230 |
| nmppsOr3V_ | 231 |
| nmppsOr4V_ | 232 |
| nmppsXorC | 233 |
| nmppsXor | 234 |
| nmppsMaskV_ | 235 |
| nmppsRShiftC | 236 |
| nmppsRShiftC_ | 237 |
| nmppsRShiftC_AddC_ | 238 |
| nmppsDisplaceBits | 239 |
| Операции сравнения | 198 |
| nmppsMax_ | 251 |
| nmppsMin | 253 |
| nmppsMaxIndx_ | 255 |
| nmppsMinIndx_ | 257 |
| nmppsMinIndxVN_ | 259 |
| nmppsFirstZeroIndx | 260 |
| nmppsFirstNonZeroIndx | 261 |
| nmppsLastZeroIndx | 262 |
| nmppsLastNonZeroIndx | 263 |
| nmppsMinEvery_ | 264 |
| nmppsMaxEvery_ | 265 |
| nmppsMinCmpLtV_ | 267 |
| nmppsCmpLt0 | 269 |
| nmppsCmpEq0 | 270 |
| nmppsCmpMinMaxV_ | 272 |
| nmppsClipPowC_ | 274 |
| nmppsClipCC_ | 275 |
| nmppsClipRShiftConvert_AddC_ | 276 |
| nmppsClipConvert_AddC_ | 277 |
| nmppsCmpEqC | 279 |
| nmppsCmpNe0 | 280 |

| | |
|---|-----|
| nmppsCmpNeC | 281 |
| nmppsCmpEqV_ | 284 |
| nmppsCmpNeV_ | 285 |
| Переупорядочивание и сортировка | 199 |
| VEC_QSort | 290 |
| nmppsRemap_ | 291 |
| nmppSplit | 293 |
| nmppSplitEco | 294 |
| Матричные функции | 297 |
| Инициализация и копирование | 112 |
| nmppmCopyua | 113 |
| MTR_Copyua | 114 |
| MTR_Copy | 115 |
| Функции поддержки | 126 |
| MTR_Malloc | 121 |
| MTR_Free | 122 |
| MTR_Addr | 123 |
| MTR_SetVal | 124 |
| MTR_GetVal | 125 |
| Векторно-матричные операции | 127 |
| nmppmMul_mm | 116 |
| nmppmMul_mv_ | 118 |
| nmppmMul_mv__AddC | 119 |
| MTR_ProdUnitV | 120 |
| Функции обработки сигналов | 298 |
| Свертка | 156 |
| SIG_XCorr | 161 |
| Масочная фильтрация | 157 |
| SIG_Median3 | 163 |
| КИХ-фильтрация | 164 |
| nmppsFIR_Xs | 165 |
| nmppsFIRInit_Xs | 166 |
| nmppsFIRInitAlloc_Xs | 167 |
| nmppsFIRGetStateSize_Xs | 168 |
| nmppsFIRFree | 169 |
| Изменение размеров | 158 |
| SIG_ResampleDown2 | 170 |
| SIG_ResampleUp3Down2 | 171 |
| SIG_CreateResample | 172 |
| SIG_SetResample | 173 |
| SIG_Resample_perf | 174 |
| Быстрое преобразование Фурье | 159 |
| DFT-8 | 17 |
| FFT-16 | 19 |
| FFT-32 | 21 |
| FFT-64 | 23 |
| FFT-128 | 25 |
| FFT-256 | 27 |
| FFT-512 | 29 |
| FFT-1024 | 31 |
| FFT-2048 | 33 |
| IDFT-8 | 35 |
| IFFT-16 | 36 |
| IFFT-32 | 38 |
| IFFT-64 | 40 |
| IFFT-128 | 42 |
| IFFT-256 | 44 |

| | |
|--|-----|
| IFFT-512 | 46 |
| IFFT-1024 | 48 |
| IFFT-2048 | 50 |
| FFT-Common | 52 |
| IFFT-Common | 55 |
| Функции обработки изображений | 299 |
| Floodfill | 92 |
| Переупорядочивание изображений | 100 |
| Блочное переупорядочивание | 101 |
| IMG_SplitIntoBlocks | 102 |
| IMG_MergeFromBlocks | 103 |
| Арифметические действия | 106 |
| Масочная фильтрация | 107 |
| КИХ-фильтрация | 91 |
| Инициализация и копирование | 108 |
| IMG_Convert | 98 |
| IMG_RGB32ToGray | 99 |
| Функции поддержки | 109 |
| IMG_Free | 104 |
| IMG_Release | 105 |
| Функции графического вывода текста | 110 |
| Скалярные функции | 300 |
| Инициализация | 84 |
| nmppsRand | 82 |
| Integer operations | 85 |
| nmppcSqrt | 83 |
| Fix-point 64 | 86 |
| nmppcFixSqrt64 | 75 |
| nmppcDoubleToFix64 | 76 |
| nmppcFix64ToDouble | 77 |
| nmppcFixDiv64 | 78 |
| nmppcFixSinCos64 | 79 |
| nmppcFixArcTan64 | 80 |
| Fix-point 32 | 87 |
| nmppcFixExp32 | 62 |
| nmppcFixSinCos32 | 63 |
| nmppcFixArcTan32 | 64 |
| nmppcDoubleToFix32 | 65 |
| nmppcFix32ToDouble | 66 |
| nmppcFixSqrt32 | 67 |
| nmppcFixMul32 | 68 |
| nmppcFixInv32 | 69 |
| nmppcTblFixArcSin32 | 70 |
| nmppcTblFixArcCos32 | 71 |
| nmppcTblFixCos32 | 72 |
| nmppcTblFixSin32 | 73 |
| nmppcFixDivMod32 | 74 |
| nmppcFix64Exp01 | 81 |
| Арифметические операции | 88 |
| nmppcDivC | 60 |
| nmppcProdC | 61 |
| Функции деинтерлейсинга | 89 |
| Базовые регистровые функции библиотеки | 302 |
| Элементарные функции | 305 |
| Vec_0_sub_data | 308 |
| Vec_activate_data | 309 |

| | |
|--|-----|
| Vec_activate_data_add_0 | 310 |
| Vec_activate_data_xor_data | 311 |
| Vec_activate_data_add_ram | 312 |
| Vec_afifo | 314 |
| Vec_data | 315 |
| Vec_data_add_ram | 317 |
| Vec_data_and_ram | 318 |
| Vec_data_or_ram | 319 |
| Vec_data_sub_ram | 320 |
| Vec_data_xor_ram | 321 |
| Vec_And | 325 |
| Vec_Mask | 326 |
| Vec_Or | 327 |
| Vec_Xor | 328 |
| Vec_Add | 330 |
| Vec_Sub | 345 |
| Vec_not_data | 351 |
| Vec_ram | 352 |
| Vec_ram_sub_data | 353 |
| Vec_vsum_activate_data_0 | 354 |
| функции взвешенного суммирования | 306 |
| Vec_ClipMul2D2W8_AddVr | 332 |
| Vec_ClipMulNDNW2_AddVr | 333 |
| Vec_ClipMulNDNW4_AddVr | 334 |
| Vec_ClipMulNDNW8_AddVr | 335 |
| Vec_Mul2D2W1_AddVr | 337 |
| Vec_Mul2D2W2_AddVr | 338 |
| Vec_Mul2D2W4_AddVr | 339 |
| Vec_Mul2D2W8_AddVr | 340 |
| Vec_Mul3D3W2_AddVr | 341 |
| Vec_Mul3D3W8_AddVr | 342 |
| Vec_Mul4D4W2_AddVr | 343 |
| Vec_MulVN_AddVN | 344 |
| Vec_vsum_data_0 | 355 |
| Vec_vsum_data_vr | 357 |
| Vec_vsum_shift_data_0 | 358 |
| Vec_vsum_shift_data_vr | 359 |
| Vec_vsum_shift_data_afifo | 360 |
| Целевые функции | 307 |
| Vec_Add_VV_shift | 313 |
| Vec_data_add_afifo | 316 |
| Vec_FilterCoreRow2 | 322 |
| Vec_FilterCoreRow4 | 323 |
| Vec_FilterCoreRow8 | 324 |
| Vec_Abs | 329 |
| Vec_ClipExt | 331 |
| Vec_IncNeg | 336 |
| Vec_SubAbs | 346 |
| Vec_SubVN_Abs | 347 |
| Vec_Swap | 348 |
| Vec_MUL_2V4toW8_shift | 349 |
| Vec_MUL_2V8toW16_shift | 350 |
| Vec_vsum_data_afifo | 356 |
| Vec_CompareMinV | 361 |
| Vec_CompareMaxV | 362 |
| Vec_DupValueInVector8 | 363 |
| Vec_DupValueInVector16 | 364 |
| Vec_BuildDiagWeights8 | 365 |

| | |
|----------------------------------|-----|
| Vec_BuildDiagWeights16 | 366 |
| Vec_MaxVal_v8nm8s | 367 |
| Vec_MaxVal_v4nm16s | 368 |
| Vec_MaxVal | 369 |
| Vec_MinVal_v8nm8s | 370 |
| Vec_MinVal_v4nm16s | 371 |
| Vec_MinVal | 372 |
| Vec_AccMul1D1W32_AddVr | 373 |
| контроль переполнения | 303 |

Глава 3

Иерархический список классов

3.1 Иерархия классов

Иерархия классов.

| | |
|---|-----|
| C_Allocator32 | 376 |
| C_MultiHeap | 380 |
| C_BoxImg< T > | 377 |
| C_BoxVec< T > | 377 |
| C_Heap | 378 |
| C_Img< T > | 379 |
| C_RingBuffer< T > | 384 |
| C_RingBufferRemote< T > | 386 |
| C_WarpImg< T > | 387 |
| CIMG_FIR< nmbits_in, nmbits_out > | 387 |
| ds_struct | 390 |
| I_2DSubPixelMinPosition | 391 |
| C_2DSubPixelMinPosition | 375 |
| C_2DTrigSubPixelMinPosition | 375 |
| I_PlessyCornerDetector | 391 |
| C_PlessyCornerDetector | 381 |
| C_PlessyCornerDetector_16s | 382 |
| C_PlessyCornerDetector_32f | 383 |
| int15in16x4 | 392 |
| int30in32x2 | 392 |
| int31in32x2 | 392 |
| mtr< T > | 393 |
| nm16sc | 394 |
| nmchar | 395 |
| nmchar1D< N > | 395 |
| nmchar2D< Y, X > | 396 |
| nmintpack< T > | 396 |
| nmmttr< T > | 397 |
| nmmttrpack< T > | 398 |
| NmppiFFTSpec_32fc | 400 |
| NmppsFFTSpec | 400 |
| NmppsFFTSpec_32fc | 401 |
| NmppsFrame_16s | 401 |
| NmppsFrame_16u | 401 |

| | |
|-------------------------------------|-----|
| NmppsFrame_32s | 402 |
| NmppsFrame_32u | 402 |
| NmppsFrame_64s | 402 |
| NmppsFrame_64u | 403 |
| NmppsFrame_8s | 403 |
| NmppsFrame_8u | 403 |
| NmppsMallocSpec | 404 |
| NmppsTmpSpec | 404 |
| nmreg | 405 |
| nmshort | 405 |
| nmshort2D< Y, X > | 406 |
| nmvecpack< T > | 406 |
| RGB24_nm8u | 408 |
| RGB32_nm10s | 408 |
| RGB32_nm10u | 408 |
| RGB32_nm8s | 409 |
| RGB32_nm8u | 409 |
| RGB64_nm16u | 410 |
| RPoint | 410 |
| S_BufferInfo | 410 |
| S_IMG_FilterKernel | 411 |
| S_IMG_FilterKernel_32s32s | 412 |
| s_int32x2 | 412 |
| s_nm32fc | 412 |
| s_nm32fcr | 413 |
| s_nm32sc | 413 |
| s_nm64sc | 413 |
| s_v16nm16s | 414 |
| s_v16nm16u | 414 |
| s_v16nm32s | 415 |
| s_v16nm32u | 415 |
| s_v16nm4u | 416 |
| s_v16nm8s | 416 |
| s_v16nm8u | 417 |
| s_v2nm32s | 417 |
| s_v2nm32u | 417 |
| s_v4nm16s | 418 |
| s_v4nm16u | 418 |
| s_v4nm32s | 419 |
| s_v4nm32u | 419 |
| s_v4nm8u | 419 |
| s_v8nm16s | 420 |
| s_v8nm16u | 420 |
| s_v8nm32s | 421 |
| s_v8nm32u | 421 |
| s_v8nm8s | 421 |
| s_v8nm8u | 422 |
| SpecTmp1 | 422 |
| spot_struct | 423 |
| tagSegmentInfo | 423 |
| tfixpoint< T, point > | 424 |
| Tmp2BuffSpec | 425 |
| uint16ptr | 425 |
| uint8ptr | 426 |
| v16nm4s | 427 |
| v4nm8s | 427 |
| vec< T > | 427 |

Глава 4

Алфавитный указатель классов

4.1 Классы

Классы с их кратким описанием.

| | |
|---|-----|
| C_2DSubPixelMinPosition | 375 |
| C_2DTrigSubPixelMinPosition | 375 |
| C_Allocator32 | 376 |
| C_BoxImg< T > | 377 |
| C_BoxVec< T > | 377 |
| C_Heap | |
| класс - куча | 378 |
| C_Img< T > | 379 |
| C_MultiHeap | 380 |
| C_PlessyCornerDetector | 381 |
| C_PlessyCornerDetector_16s | 382 |
| C_PlessyCornerDetector_32f | 383 |
| C_RingBuffer< T > | 384 |
| C_RingBufferRemote< T > | 386 |
| C_WarpImg< T > | 387 |
| CIMG_FIR< nmbits_in, nmbits_out > | 387 |
| ds_struct | 390 |
| I_2DSubPixelMinPosition | 391 |
| I_PlessyCornerDetector | 391 |
| int15in16x4 | 392 |
| int30in32x2 | 392 |
| int31in32x2 | 392 |
| mtr< T > | 393 |
| nm16sc | 394 |
| nmchar | 395 |
| nmchar1D< N > | 395 |
| nmchar2D< Y, X > | 396 |
| nmintpack< T > | 396 |
| nmmtr< T > | 397 |
| nmmtrpack< T > | 398 |
| NmppiFFTSpec_32fc | 400 |
| NmppsFFTSpec | 400 |
| NmppsFFTSpec_32fc | 401 |
| NmppsFrame_16s | 401 |
| NmppsFrame_16u | 401 |

| | |
|---|-----|
| NmppsFrame_32s | 402 |
| NmppsFrame_32u | 402 |
| NmppsFrame_64s | 402 |
| NmppsFrame_64u | 403 |
| NmppsFrame_8s | 403 |
| NmppsFrame_8u | 403 |
| NmppsMallocSpec | 404 |
| NmppsTmpSpec | 404 |
| nmreg | 405 |
| nmshort | 405 |
| nmshort2D< Y, X > | 406 |
| nmvecpack< T > | 406 |
| RGB24_nm8u | 408 |
| RGB32_nm10s | 408 |
| RGB32_nm10u | 408 |
| RGB32_nm8s | 409 |
| RGB32_nm8u | 409 |
| RGB64_nm16u | 410 |
| RPoint | 410 |
| S_BufferInfo | |
| класс буфер - заголовок в начале выделяемой динамической памяти | 410 |
| S_IMG_FilterKernel | 411 |
| S_IMG_FilterKernel_32s32s | 412 |
| s_int32x2 | 412 |
| s_nm32fc | 412 |
| s_nm32fcr | 413 |
| s_nm32sc | 413 |
| s_nm64sc | 413 |
| s_v16nm16s | 414 |
| s_v16nm16u | 414 |
| s_v16nm32s | 415 |
| s_v16nm32u | 415 |
| s_v16nm4u | 416 |
| s_v16nm8s | 416 |
| s_v16nm8u | 417 |
| s_v2nm32s | 417 |
| s_v2nm32u | 417 |
| s_v4nm16s | 418 |
| s_v4nm16u | 418 |
| s_v4nm32s | 419 |
| s_v4nm32u | 419 |
| s_v4nm8u | 419 |
| s_v8nm16s | 420 |
| s_v8nm16u | 420 |
| s_v8nm32s | 421 |
| s_v8nm32u | 421 |
| s_v8nm8s | 421 |
| s_v8nm8u | 422 |
| SpecTmp1 | 422 |
| spot_struct | 423 |
| tagSegmentInfo | 423 |
| tfixpoint< T, point > | 424 |
| Tmp2BuffSpec | 425 |
| uint16ptr | 425 |
| uint8ptr | 426 |
| v16nm4s | 427 |
| v4nm8s | 427 |
| vec< T > | 427 |

Глава 5

Список файлов

5.1 Файлы

Полный список документированных файлов.

| | |
|--|-----|
| D:/GIT/nmpp/include/crtdbg2.h | ?? |
| D:/GIT/nmpp/include/fft.h | ?? |
| D:/GIT/nmpp/include/fft_32fc.h | ?? |
| D:/GIT/nmpp/include/fftexp.h | ?? |
| D:/GIT/nmpp/include/malloc32.h | ?? |
| D:/GIT/nmpp/include/metric.h | ?? |
| D:/GIT/nmpp/include/minrep.h | ?? |
| D:/GIT/nmpp/include/multiheap.h | ?? |
| D:/GIT/nmpp/include/nmchar.h | ?? |
| D:/GIT/nmpp/include/nmdef.h | ?? |
| D:/GIT/nmpp/include/nmplc.h | ?? |
| D:/GIT/nmpp/include/nmpli.h | ?? |
| D:/GIT/nmpp/include/nmplm.h | ?? |
| D:/GIT/nmpp/include/nmpls.h | ?? |
| D:/GIT/nmpp/include/nmplv.h | ?? |
| D:/GIT/nmpp/include/nmpp.h | ?? |
| D:/GIT/nmpp/include/nmshort.h | ?? |
| D:/GIT/nmpp/include/nmtl.h | ?? |
| D:/GIT/nmpp/include/nmtype.h | 431 |
| D:/GIT/nmpp/include/ownmalloc.h | ?? |
| D:/GIT/nmpp/include/ringbuffer.h | ?? |
| D:/GIT/nmpp/include/ringremote.h | ?? |
| D:/GIT/nmpp/include/tfixpoint.h | ?? |
| D:/GIT/nmpp/include/nmplc/cArithmetic.h | ?? |
| D:/GIT/nmpp/include/nmplc/cfixpnt32.h | ?? |
| D:/GIT/nmpp/include/nmplc/cfixpnt64.h | ?? |
| D:/GIT/nmpp/include/nmplc/cInit.h | ?? |
| D:/GIT/nmpp/include/nmplc/cInteger.h | ?? |
| D:/GIT/nmpp/include/nmplc/nmplc.h | ?? |
| D:/GIT/nmpp/include/nmpli/filter.h | ?? |
| D:/GIT/nmpp/include/nmpli/iArithmetics.h | ?? |
| D:/GIT/nmpp/include/nmpli/iCellTexture.h | ?? |
| D:/GIT/nmpp/include/nmpli/iDef.h | ?? |
| D:/GIT/nmpp/include/nmpli/iDeinterlace.h | ?? |
| D:/GIT/nmpp/include/nmpli/iFilter.h | ?? |

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

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

Глава 6

Группы

6.1 DFT-8

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

Функции

- void [nmppsDFT8Fwd_32fc](#) ([nm32fcr](#) *x, [nm32fcr](#) *X, [NmppsFFTSpec_32fc](#) *spec)

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

6.1.1 Подробное описание

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

6.1.2 Функции

6.1.2.1 nmppsDFT8Fwd_32fc()

```
void nmppsDFT8Fwd_32fc (  
    nm32fcr * x,  
    nm32fcr * X,  
    NmppsFFTSpec\_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.2 FFT-16

Функции

- void `nmppsFFT16Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 16 комплексных чисел
- int `nmppsFFT16FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-16.

6.2.1 Подробное описание

6.2.2 Функции

6.2.2.1 `nmppsFFT16Fwd_32fc()`

```
void nmppsFFT16Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.2.2.2 `nmppsFFT16FwdInitAlloc_32fc()`

```
int nmppsFFT16FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.3 FFT-32

Функции

- void `nmppsFFT32Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 32 комплексных чисел
- int `nmppsFFT32FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-32.

6.3.1 Подробное описание

6.3.2 Функции

6.3.2.1 `nmppsFFT32Fwd_32fc()`

```
void nmppsFFT32Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.3.2.2 `nmppsFFT32FwdInitAlloc_32fc()`

```
int nmppsFFT32FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.4 FFT-64

Функции

- void `nmppsFFT64Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 64 комплексных чисел
- int `nmppsFFT64FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-64.

6.4.1 Подробное описание

6.4.2 Функции

6.4.2.1 `nmppsFFT64Fwd_32fc()`

```
void nmppsFFT64Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.4.2.2 `nmppsFFT64FwdInitAlloc_32fc()`

```
int nmppsFFT64FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.5 FFT-128

Функции

- void `nmppsFFT128Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 128 комплексных чисел
- int `nmppsFFT128FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-128.

6.5.1 Подробное описание

6.5.2 Функции

6.5.2.1 `nmppsFFT128Fwd_32fc()`

```
void nmppsFFT128Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.5.2.2 `nmppsFFT128FwdInitAlloc_32fc()`

```
int nmppsFFT128FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.6 FFT-256

Функции

- void `nmppsFFT256Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 256 комплексных чисел
- int `nmppsFFT256FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-256.

6.6.1 Подробное описание

6.6.2 Функции

6.6.2.1 `nmppsFFT256Fwd_32fc()`

```
void nmppsFFT256Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.6.2.2 `nmppsFFT256FwdInitAlloc_32fc()`

```
int nmppsFFT256FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.7 FFT-512

Функции

- void `nmppsFFT512Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 512 комплексных чисел
- int `nmppsFFT512FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-512.

6.7.1 Подробное описание

6.7.2 Функции

6.7.2.1 `nmppsFFT512Fwd_32fc()`

```
void nmppsFFT512Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.7.2.2 `nmppsFFT512FwdInitAlloc_32fc()`

```
int nmppsFFT512FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.8 FFT-1024

Функции

- void `nmppsFFT1024Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 1024 комплексных чисел
- int `nmppsFFT1024FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-1024.

6.8.1 Подробное описание

6.8.2 Функции

6.8.2.1 `nmppsFFT1024Fwd_32fc()`

```
void nmppsFFT1024Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.8.2.2 `nmppsFFT1024FwdInitAlloc_32fc()`

```
int nmppsFFT1024FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.9 FFT-2048

Функции

- void `nmppsFFT2048Fwd_32fc` (`nm32fcr *x`, `nm32fcr *X`, `NmppsFFTSpec_32fc *spec`)
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 2048 комплексных чисел
- int `nmppsFFT2048FwdInitAlloc_32fc` (`NmppsFFTSpec_32fc **addr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-2048.

6.9.1 Подробное описание

6.9.2 Функции

6.9.2.1 `nmppsFFT2048Fwd_32fc()`

```
void nmppsFFT2048Fwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.9.2.2 `nmppsFFT2048FwdInitAlloc_32fc()`

```
int nmppsFFT2048FwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** addr )
```

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

Аргументы

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

Возвращает

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

6.10 IDFT-8

Функции

- void `nmppsDFT8Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)

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

6.10.1 Подробное описание

6.10.2 Функции

6.10.2.1 `nmppsDFT8Inv_32fc()`

```
void nmppsDFT8Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.11 IFFT-16

Функции

- void `nmppsFFT16Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 16 комплексных чисел
- int `nmppsFFT16InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-16.

6.11.1 Подробное описание

6.11.2 Функции

6.11.2.1 `nmppsFFT16Inv_32fc()`

```
void nmppsFFT16Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.11.2.2 `nmppsFFT16InvInitAlloc_32fc()`

```
int nmppsFFT16InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```

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

Аргументы

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

Возвращает

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

6.12 IFFT-32

Функции

- void `nmppsFFT32Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 32 комплексных чисел
- int `nmppsFFT32InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-32.

6.12.1 Подробное описание

6.12.2 Функции

6.12.2.1 `nmppsFFT32Inv_32fc()`

```
void nmppsFFT32Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.12.2.2 `nmppsFFT32InvInitAlloc_32fc()`

```
int nmppsFFT32InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```


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

Аргументы

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

Возвращает

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

6.13 IFFT-64

Функции

- void `nmppsFFT64Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 64 комплексных чисел
- int `nmppsFFT64InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-64.

6.13.1 Подробное описание

6.13.2 Функции

6.13.2.1 `nmppsFFT64Inv_32fc()`

```
void nmppsFFT64Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.13.2.2 `nmppsFFT64InvInitAlloc_32fc()`

```
int nmppsFFT64InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```

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

Аргументы

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

Возвращает

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

6.14 IFFT-128

Функции

- void `nmppsFFT128Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 128 комплексных чисел
- int `nmppsFFT128InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-128.

6.14.1 Подробное описание

6.14.2 Функции

6.14.2.1 `nmppsFFT128Inv_32fc()`

```
void nmppsFFT128Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.14.2.2 `nmppsFFT128InvInitAlloc_32fc()`

```
int nmppsFFT128InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```

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

Аргументы

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

Возвращает

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

6.15 IFFT-256

Функции

- void `nmppsFFT256Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 256 комплексных чисел
- int `nmppsFFT256InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-256.

6.15.1 Подробное описание

6.15.2 Функции

6.15.2.1 `nmppsFFT256Inv_32fc()`

```
void nmppsFFT256Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.15.2.2 `nmppsFFT256InvInitAlloc_32fc()`

```
int nmppsFFT256InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```

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

Аргументы

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

Возвращает

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

6.16 IFFT-512

Функции

- void `nmppsFFT512Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 512 комплексных чисел
- int `nmppsFFT512InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-512.

6.16.1 Подробное описание

6.16.2 Функции

6.16.2.1 `nmppsFFT512Inv_32fc()`

```
void nmppsFFT512Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.16.2.2 `nmppsFFT512InvInitAlloc_32fc()`

```
int nmppsFFT512InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```


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

Аргументы

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

Возвращает

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

6.17 IFFT-1024

Функции

- void `nmppsFFT1024Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 1024 комплексных чисел
- int `nmppsFFT1024InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-1024.

6.17.1 Подробное описание

6.17.2 Функции

6.17.2.1 `nmppsFFT1024Inv_32fc()`

```
void nmppsFFT1024Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.17.2.2 `nmppsFFT1024InvInitAlloc_32fc()`

```
int nmppsFFT1024InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```

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

Аргументы

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

Возвращает

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

6.18 IFFT-2048

Функции

- void `nmppsFFT2048Inv_32fc` (`nm32fcr *ix`, `nm32fcr *iX`, `NmppsFFTSpec_32fc *ispec`)
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 2048 комплексных чисел
- int `nmppsFFT2048InvInitAlloc_32fc` (`NmppsFFTSpec_32fc **iaddr`)
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-2048.

6.18.1 Подробное описание

6.18.2 Функции

6.18.2.1 `nmppsFFT2048Inv_32fc()`

```
void nmppsFFT2048Inv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * ispec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.18.2.2 `nmppsFFT2048InvInitAlloc_32fc()`

```
int nmppsFFT2048InvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iaddr )
```

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

Аргументы

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

Возвращает

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

6.19 FFT-Common

Функции

- `int nmppsFFTFwd_32fc (nm32fcr *x, nm32fcr *X, NmppsFFTSpec_32fc *Spec)`
Функция для вычисления прямого БПФ с плавающей точкой над вектором длины от 8 до 2048.
- `int nmppsFFTFwdInitAlloc_32fc (NmppsFFTSpec_32fc **Spec, int Order)`
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ с плавающей точкой над вектором длины от 8 до 2048.

6.19.1 Подробное описание

6.19.2 Функции

6.19.2.1 nmppsFFTFwd_32fc()

```
int nmppsFFTFwd_32fc (
    nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fc * Spec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.19.2.2 nmppsFFTFwdInitAlloc_32fc()

```
int nmppsFFTFwdInitAlloc_32fc (
    NmppsFFTSpec_32fc ** Spec,
    int Order )
```

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

Аргументы

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

Возвращает

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

6.20 IFFT-Common

Функции

- `int nmppsFFTInv_32fc (nm32fcr *ix, nm32fcr *iX, NmppsFFTSpec_32fc *iSpec)`
Функция для вычисления обратного БПФ с плавающей точкой над вектором длины от 8 до 2048.
- `int nmppsFFTInvInitAlloc_32fc (NmppsFFTSpec_32fc **iSpec, int iOrder)`
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ с плавающей точкой над вектором длины от 8 до 2048.

6.20.1 Подробное описание

6.20.2 Функции

6.20.2.1 nmppsFFTInv_32fc()

```
int nmppsFFTInv_32fc (
    nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fc * iSpec )
```

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

Аргументы

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

Возвращаемые значения

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

Аргументы

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

6.20.2.2 nmppsFFTInvInitAlloc_32fc()

```
int nmppsFFTInvInitAlloc_32fc (
    NmppsFFTSpec_32fc ** iSpec,
    int iOrder )
```

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

Аргументы

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

Возвращает

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

6.21 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)`

6.21.1 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()`.

6.22 nmppsFree

Освобождение памяти для векторов.

Функции

- void nmppsFree (void *ptr)

6.22.1 Подробное описание

Освобождение памяти для векторов.

Заметки

Данная функция должна вызываться только для векторов, распределенных с помощью функций nmppsMalloc_.

6.23 nmppcDivC

частное двух комплексных чисел

Функции

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

6.23.1 Подробное описание

частное двух комплексных чисел

<>

Аргументы

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

6.24 nmppcProdC

произведение двух комплексных чисел.

Функции

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

6.24.1 Подробное описание

произведение двух комплексных чисел.

Аргументы

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

6.25 nmppsFixExp32

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

Функции

- int nmppsFixExp32 (int nVal)

6.25.1 Подробное описание

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

Аргументы

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

Возвращает

Экспонента числа в формате с фиксированной точкой (16.16)

6.26 nmppcFixSinCos32

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

Функции

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

6.26.1 Подробное описание

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

Аргументы

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

Возвращаемые значения

| | |
|-------|----------------------|
| pnSin | указатель на синус |
| pnCos | указатель на косинус |

6.27 nmppsFixArcTan32

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

Функции

- `int nmppsFixArcTan32 (int nArg)`

6.27.1 Подробное описание

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

Аргументы

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

Возвращает

Арктангенс

6.28 nmppcDoubleToFix32

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

Функции

- `int nmppcDoubleToFix32 (double arg)`

6.28.1 Подробное описание

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

Аргументы

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

Возвращает

Число с фиксированной точкой

6.29 nmppcFix32ToDouble

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

Функции

- double nmppcFix32ToDouble (int arg)

6.29.1 Подробное описание

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

Аргументы

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

Возвращает

Число с плавающей точкой

6.30 nmppcFixSqrt32

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

Функции

- unsigned int nmppcFixSqrt32 (unsigned int nVal)

6.30.1 Подробное описание

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

Аргументы

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

Возвращает

Квадратный корень в формате с фиксированной точкой (16.16)

6.31 nmppcFixMul32

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

Функции

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

6.31.1 Подробное описание

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

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

Аргументы

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

Возвращает

Произведение в формате с фиксированной точкой (16.16)

Аргументы

| | |
|----|----------|
| nX | Делимое |
| nY | Делитель |

Возвращает

Частное от деления в формате с фиксированной точкой (16.16)

6.32 nmppcFixInv32

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

Функции

- int nmppcFixInv32 (int nVal, int nFixpoint)

6.32.1 Подробное описание

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

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

\param nVal

Делитель

Аргументы

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

Возвращает

Частное от деления нав формате с фиксированной точкой (16.16)

6.33 nmppsTblFixArcSin32

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

Функции

- `int nmppsTblFixArcSin32 (int nArg)`

6.33.1 Подробное описание

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

Аргументы

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

Возвращает

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

6.34 nmppcTblFixArcCos32

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

Функции

- `int nmppcTblFixArcCos32 (int nArg)`

6.34.1 Подробное описание

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

Аргументы

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

Возвращает

Угол в диапазоне от 0 до π в формате `fixed-point (16.16)`

6.35 nmppsTblFixCos32

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

Функции

- `int nmppsTblFixCos32 (int nArg)`

6.35.1 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 π в формате fixed-point (16.16) |
|------|--|

Возвращает

значение \cos в формате fixed-point (16.16)

6.36 nmppcTblFixSin32

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

Функции

- `int nmppcTblFixSin32 (int nArg)`

6.36.1 Подробное описание

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

Аргументы

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

Возвращает

значение \sin в формате fixed-point (16.16)

6.37 nmppcFixDivMod32

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

Функции

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

6.37.1 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) |

6.38 nmppcFixSqrt64

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

Функции

- unsigned long nmppcFixSqrt64 (unsigned long x)

6.38.1 Подробное описание

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

Аргументы

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

Возвращает

Квадратный корень в формате с фиксированной точкой (32.32)

6.39 nmppcDoubleToFix64

Функция перевода из Fixed-Point 64 в Double.

Функции

- long nmppcDoubleToFix64 (double arg, int fixpoint)

6.39.1 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

6.40 nmppcFix64ToDouble

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

Функции

- double nmppcFix64ToDouble (long arg, int fixpoint)

6.40.1 Подробное описание

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

Аргументы

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

Возвращает

Число с плавающей точкой

6.41 nmppcFixDiv64

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

Функции

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

6.41.1 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 |
|-----------|--|

6.42 nmppcFixSinCos64

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

Функции

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

6.42.1 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 |

6.43 nmppsFixArcTan64

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

Функции

- long nmppsFixArcTan64 (long nArg)

6.43.1 Подробное описание

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

Аргументы

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

Арктангенс

6.44 nmppcFix64Exp01

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

Функции

- long nmppcFix64Exp01 (long nArg)

6.44.1 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)

6.45 nmppsRand

Генерация случайного числа с равномерным распределением.

Функции

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

6.45.1 Подробное описание

Генерация случайного числа с равномерным распределением.

Аргументы

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

Возвращает

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

6.46 nmppcSqrt

Вычисление квадратного корня

Функции

- unsigned int nmppcSqrt_64u (unsigned long long x)

6.46.1 Подробное описание

Вычисление квадратного корня

Аргументы

| | |
|---|---------------|
| x | Входное число |
|---|---------------|

Возвращает

Квадратный корень

6.47 Инициализация

Группы

- [nmppsRand](#)

Генерация случайного числа с равномерным распределением.

6.47.1 Подробное описание

6.48 Integer operations

Группы

- [nmppsSqrt](#)

Вычисление квадратного корня

6.48.1 Подробное описание

6.49 Fix-point 64

Группы

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

6.49.1 Подробное описание

6.50 Fix-point 32

Группы

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

6.50.1 Подробное описание

6.51 Арифметические операции

Группы

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

6.51.1 Подробное описание

6.52 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)

6.52.1 Подробное описание

6.52.2 Функции

6.52.2.1 IMG_DeinterlaceBlend()

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

Функция объединяет два полукдра в один полный кадр.

Аргументы

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

Возвращает

void

6.52.2.2 IMG_DeinterlaceSplit()

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

Функция разделяет кадр на два полукдра.

Аргументы

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

Возвращает

void

6.53 КИХ-фильтрация

Двумерная КИХ фильтрация

Классы

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

6.53.1 Подробное описание

Двумерная КИХ фильтрация

6.54 Floodfill

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

Функции

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

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

6.54.1 Подробное описание

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

Аргументы

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

Возвращаемые значения

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

Возвращает

Число сегментов на изображении

Заметки

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

6.54.2 Функции

6.54.2.1 FloodFill8()

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

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

Аргументы

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

Аргументы

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

Аргументы

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

Аргументы

| | |
|--------|--|
| dXYmin | <p>минимальная протяженность пятна по X или по Y в пикселах,</p> <ul style="list-style-type: none"> при IDropSpot=1: пятно отбраковывается, если его протяженность меньше dXYMax, при IDropSpot=0 проверок не производится, но параметр используется для определения шага поиска по строкам входной матрицы src (и начальной строки поиска): $dI = \max(1, dXYmin)$ <p>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 параметра IDropSpot надо задавать dXYmin, по возможности, большим.</p> |
| dXYmax | <p>максимальная протяженность пятна по X или по Y в пикселах,</p> <ul style="list-style-type: none"> при IDropSpot=1: пятно отбраковывается, если его протяженность больше dXYMax, при IDropSpot=0 проверок не производится, параметр можно задать 0. <p>Ограничения: nPxMin, nPxMax, dXYmin, dXYmax - неотрицательные, (nPxMin <= nPxMax), (dXYmin <= dXYmax).</p> |

Возвращает

Функция возвращает целое число:

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

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

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

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

```

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

```

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

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

Заметки

Предельные случаи

Следует выделить предельные случаи:

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

Алгоритм

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

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

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

2. начало обработки (очередного) пятна:

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

3. while-цикл обработки пятна:

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

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

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

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

Меняется указатель `noPx1=noPx11`.

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

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

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

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

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

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

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

4. Конец программы.

6.55 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)

6.55.1 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;

6.56 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)

6.56.1 Подробное описание

Преобразование пикселей из RGB в яркость

Аргументы

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

6.57 Переупорядочивание изображений

Группы

- [Блочное переупорядочивание](#)

Блочное перупорядочивание изображений.

6.57.1 Подробное описание

6.58 Блочное переупорядочивание

Блочное переупорядочивание изображений.

Группы

- [IMG_SplitIntoBlocks](#)
Преобразует изображение в последовательность квадратных блоков.
- [IMG_MergeFromBlocks](#)
Объединяет последовательность квадратных блоков в изображение.

6.58.1 Подробное описание

Блочное переупорядочивание изображений.

6.59 IMG_SplitIntoBlocks

Преобразует изображение в последовательность квадратных блоков.

Функции

- void IMG_SplitIntoBlocks8x8 (nm8s *pSrcImg, nm8s *pDstBlockSeq, int nWidth, int nHeight)

6.59.1 Подробное описание

Преобразует изображение в последовательность квадратных блоков.

Аргументы

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

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

Выходная последовательность блоков имеет вид

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


6.60 IMG_MergeFromBlocks

Объединяет последовательность квадратных блоков в изображение.

Функции

- void IMG_MergeFromBlocks8x8 (nm8s *pSrcBlockSeq, nm8s *pDstImg, int nWidth, int nHeight)

6.60.1 Подробное описание

Объединяет последовательность квадратных блоков в изображение.

Аргументы

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

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

Результирующее изображение имеет вид

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

6.61 IMG_Free

Освобождение памяти для изображений.

Функции

- void IMG_Free (void *ptr)

6.61.1 Подробное описание

Освобождение памяти для изображений.

Заметки

Данная функция должна вызываться только для векторов, распределенных с помощью функций IMG_Malloc.

6.62 IMG_Release

Освобождение блоков памяти, выделенных функциями IMG_Create***.

Функции

- `__INLINE__ void IMG_ReleaseObject (nm64s *pKernel)`

6.62.1 Подробное описание

Освобождение блоков памяти, выделенных функциями IMG_Create***.

6.63 Арифметические действия

6.64 Масочная фильтрация

Группы

- [КИХ-фильтрация](#)

Двумерная КИХ фильтрация

6.64.1 Подробное описание

6.65 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

6.65.1 Подробное описание

6.66 Функции поддержки

Группы

- [IMG_Free](#)
Освобождение памяти для изображений.
- [IMG_Release](#)
Освобождение блоков памяти, выделенных функциями IMG_Create***.

6.66.1 Подробное описание

6.67 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.

6.67.1 Подробное описание

6.67.2 Функции

6.67.2.1 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

6.67.2.2 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.

6.67.2.3 IMG_Print8x15()

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

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

Аргументы

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

Возвращает

x-coordinate behind printed text

Заметки

Russian text is supported in win-1251 codepage

6.68 Инициализация и копирование

Группы

- [nmprpmCopyua](#)

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.

6.68.1 Подробное описание

6.69 nmppmCopyua

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

Функции

- void nmppmCopyua_8s (nm8s *pSrcMtr, int nSrcStride, int nSrcOffset, nm8s *pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopyua_16s (nm16s *pSrcMtr, int nSrcStride, int nSrcOffset, nm16s *pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopyua_32s (nm32s *pSrcMtr, int nSrcStride, int nSrcOffset, nm32s *pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy_32fc (double *pSrcMtr, int nSrcStride, double *pDstMtr, int nDstStride, int nHeight, int nWidth)

6.69.1 Подробное описание

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

Restrictions:

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

6.70 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 nDstOffset, int nHeight, int nWidth)
- void nmppmCopyau_16s (nm16s *pSrcMtr, int nSrcStride, nm16s *pDstMtr, int nDstStride, int nDstOffset, int nHeight, int nWidth)
- void nmppmCopyau_32s (nm32s *pSrcMtr, int nSrcStride, nm32s *pDstMtr, int nDstStride, int nDstOffset, int nHeight, int nWidth)

6.70.1 Подробное описание

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

Аргументы

| | |
|------------|--------------------------------------|
| pSrcMtr | Исходная матрица. |
| nSrcStride | Ширина исходной матрицы в элементах. |

Возвращаемые значения

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

Аргументы

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

Возвращает

void

Restrictions:

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

6.71 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)

6.71.1 Подробное описание

Функции копирования прямоугольных областей памяти между двумерными массивами.

Аргументы

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

Возвращаемые значения

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

Возвращает

void

Restrictions:

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

6.72 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)

6.72.1 Подробное описание

Умножение матрицы на матрицу.

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.73 nmppmMul_mv_

Умножение матрицы на вектор.

Функции

- void nmppmMul_mv_8s64s (nm8s *pSrcMtr, nm64s *pSrcVec, nm64s *pDstVec, int nHeight, int nWidth)
- void nmppmMul_mv_16s64s (nm16s *pSrcMtr, nm64s *pSrcVec, nm64s *pDstVec, int nHeight, int nWidth)
- void nmppmMul_mv_32s64s (nm32s *pSrcMtr, nm64s *pSrcVec, nm64s *pDstVec, int nHeight, int nWidth)
- void nmppmMul_mv_8s16s_8xH (v8nm8s *pSrcMtr, v8nm16s *pSrcVec, nm16s *pDstVec, int nHeight)
- void nmppmMul_mv_16s16s_8xH (v8nm16s *pSrcMtr, v8nm16s *pSrcVec, nm16s *pDstVec, int nHeight)
- void nmppmMul_mv_colmajor_8s64s (const nm8s *pSrcMtr, const nm64s *pSrcVec, nm64s *pDstVec, int nHeight, int nWidth)
- void nmppmMul_mv_colmajor_16s64s (const nm16s *pSrcMtr, const nm64s *pSrcVec, nm64s *pDstVec, int nHeight, int nWidth)
- void nmppmMul_mv_colmajor_32s64s (const nm32s *pSrcMtr, const nm64s *pSrcVec, nm64s *pDstVec, int nHeight, int nWidth)

6.73.1 Подробное описание

Умножение матрицы на вектор.

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.74 nmppmMul_mv__AddC

Умножение матрицы на вектор с добавлением константы.

Функции

- void nmppmMul_mv__AddC (v2nm32s *pSrcMtr, v2nm32s *pnSrcVec, int nAddVal, nm32s *pDstVec, int nHeight)

6.74.1 Подробное описание

Умножение матрицы на вектор с добавлением константы.

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.75 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)

6.75.1 Подробное описание

Умножение матрицы на единичный вектор.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.76 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.

6.77 MTR_Free

Освобождение памяти для матриц.

Функции

- `__INLINE__ void MTR_Free (void *ptr)`

6.77.1 Подробное описание

Освобождение памяти для матриц.

Заметки

Данная функция должна вызываться только для векторов, распределенных с помощью функций MTR_Malloc.

6.78 MTR_Addr

Возвращает адрес ячейки памяти, содержащей указанный элемент.

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

Функции

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

6.78.1 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.

6.79 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, uint8b nVal)`
- `__INLINE__ void MTR_SetVal_16u (nm16u *pMtr, int nWidth, int nY, int nX, uint16b nVal)`
- `__INLINE__ void MTR_SetVal_32u (nm32u *pMtr, int nWidth, int nY, int nX, uint32b nVal)`
- `__INLINE__ void MTR_SetVal_64u (nm64u *pMtr, int nWidth, int nY, int nX, uint64b nVal)`

6.79.1 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

void

6.80 MTR_GetVal

Считывает значение элемента матрицы.

Функции

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

6.80.1 Подробное описание

Считывает значение элемента матрицы.

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.81 Функции поддержки

Группы

- [MTR_Malloc](#)

Распределение памяти для матриц библиотеки.

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

- [MTR_Free](#)

Освобождение памяти для матриц.

- [MTR_Addr](#)

Возвращает адрес ячейки памяти, содержащей указанный элемент.

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

- [MTR_SetVal](#)

Записывает число в элемент матрицы.

- [MTR_GetVal](#)

Считывает значение элемента матрицы.

6.81.1 Подробное описание

6.82 Векторно-матричные операции

Группы

- [nmppmMul_mm](#)
Умножение матрицы на матрицу.
- [nmppmMul_mv_](#)
Умножение матрицы на вектор.
- [nmppmMul_mv__AddC](#)
Умножение матрицы на вектор с добавлением константы.
- [MTR_ProdUnitV](#)
Умножение матрицы на единичный вектор.

6.82.1 Подробное описание

6.83 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.

6.83.1 Подробное описание

6.83.2 Функции

6.83.2.1 FFT_Fwd256()

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

Прямое быстрое преобразование Фурье-256.

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

Аргументы

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

Возвращает

void

Заметки

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

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

6.84 IFFT-256

Функции

- void [FFT_Inv256Set6bit](#) ()
Устанавливает 6-битную точность вычислений
- void [FFT_Inv256Set7bit](#) ()
Устанавливает 7-битную точность вычислений
- void [FFT_Inv256](#) (nm32sc *GSrcBuffer, nm32sc *GDstBuffer, void *LBuffer, void *GBuffer, int ShiftR1=8, int ShiftR2=-1)
Обратное быстрое преобразование Фурье. ОБПФ-256.

6.84.1 Подробное описание

6.84.2 Функции

6.84.2.1 FFT_Inv256()

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

Обратное быстрое преобразование Фурье. ОБПФ-256.

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

Аргументы

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

Возвращает

void

Заметки

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

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

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

| | | | | | | | | | | | | |
|---|--|---|--|---|--|---|--|---|--|---|--|-------|
| G | | G | | G | | L | | 0 | | 0 | | 22.83 |
| G | | G | | G | | G | | 8 | | 0 | | 20.73 |
| G | | G | | G | | G | | 0 | | 0 | | 20.73 |

6.85 FFT-512

Функции

- void [FFT_Fwd512Set6bit](#) ()
Устанавливает 6-битную точность вычислений
- void [FFT_Fwd512Set7bit](#) ()
Устанавливает 7-битную точность вычислений
- void [FFT_Fwd512](#) (nm32sc *GSrcBuffer, nm32sc *GDstBuffer, void *LBuffer, void *GBuffer, int ShiftR=-1)
Прямое быстрое преобразование Фурье-512.

6.85.1 Подробное описание

6.85.2 Функции

6.85.2.1 FFT_Fwd512()

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

Прямое быстрое преобразование Фурье-512.

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

Аргументы

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

Возвращает

void

Заметки

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

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

6.86 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.

6.86.1 Подробное описание

6.86.2 Функции

6.86.2.1 FFT_Inv512()

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

Обратное быстрое преобразование Фурье. ОБПФ-512.

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

Аргументы

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

Возвращает

void

Заметки

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

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

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

| | | | | | | | | | | | | |
|---|--|---|--|---|--|---|--|---|--|---|--|-------|
| G | | G | | G | | L | | 0 | | 0 | | 24.81 |
| G | | G | | G | | G | | 9 | | 0 | | 28.40 |
| G | | G | | G | | G | | 0 | | 0 | | 28.39 |

6.87 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.

6.87.1 Подробное описание

6.87.2 Функции

6.87.2.1 FFT_Fwd1024()

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

Прямое быстрое преобразование Фурье-1024.

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

Аргументы

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

Возвращает

void

Заметки

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

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

6.88 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.

6.88.1 Подробное описание

6.88.2 Функции

6.88.2.1 FFT_Inv1024()

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

Обратное быстрое преобразование Фурье. ОБПФ-1024.

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

Аргументы

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

Возвращает

void

Заметки

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

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

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

| | | | | | | | | | | | | |
|---|--|---|--|---|--|---|--|----|--|---|--|-------|
| G | | G | | G | | L | | 0 | | 0 | | 28.51 |
| G | | G | | G | | G | | 10 | | 0 | | 27.46 |
| G | | G | | G | | G | | 0 | | 0 | | 27.46 |

6.89 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.

6.89.1 Подробное описание

6.89.2 Функции

6.89.2.1 FFT_Fwd2048()

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

Прямое быстрое преобразование Фурье-2048.

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

Аргументы

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

Возвращает

void

Заметки

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

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

6.90 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.

6.90.1 Подробное описание

6.90.2 Функции

6.90.2.1 FFT_Inv2048()

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

Обратное быстрое преобразование Фурье. ОБПФ-2048.

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

Аргументы

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

Возвращает

void

Заметки

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

| GSrcBuffer | LDstBuffer | LBuffer | GBuffer | ShiftR1 | ShiftR2 | clocks |
|------------|------------|---------|---------|---------|---------|--------|
| L | L | L | L | 11 | -1 | 30.58 |
| L | L | L | L | 0 | -1 | 30.58 |
| L | L | L | G | 11 | -1 | 26.52 |
| L | L | L | G | 0 | -1 | 26.52 |
| L | L | G | L | 11 | -1 | 31.33 |
| L | L | G | L | 0 | -1 | 31.33 |
| L | L | G | G | 11 | -1 | 29.26 |
| L | L | G | G | 0 | -1 | 29.26 |
| L | G | L | L | 11 | -1 | 29.56 |
| L | G | L | L | 0 | -1 | 29.55 |
| L | G | L | G | 11 | -1 | 27.48 |
| L | G | L | G | 0 | -1 | 27.48 |
| L | G | G | L | 11 | -1 | 30.30 |
| L | G | G | L | 0 | -1 | 30.30 |
| L | G | G | G | 11 | -1 | 30.22 |
| L | G | G | G | 0 | -1 | 30.22 |
| G | L | L | L | 11 | -1 | 29.56 |
| G | L | L | L | 0 | -1 | 29.56 |
| G | L | L | G | 11 | -1 | 25.49 |
| G | L | L | G | 0 | -1 | 25.49 |
| G | L | G | L | 11 | -1 | 32.29 |
| G | L | G | L | 0 | -1 | 32.29 |
| G | L | G | G | 11 | -1 | 30.22 |
| G | L | G | G | 0 | -1 | 30.22 |
| G | G | L | L | 11 | -1 | 28.53 |
| G | G | L | L | 0 | -1 | 28.53 |
| G | G | L | G | 11 | -1 | 26.46 |
| G | G | L | G | 0 | -1 | 26.46 |
| G | G | G | L | 11 | -1 | 31.26 |
| G | G | G | L | 0 | -1 | 31.26 |
| G | G | G | G | 11 | -1 | 31.19 |
| G | G | G | G | 0 | -1 | 31.18 |
| L | L | L | L | 11 | 0 | 30.58 |
| L | L | L | L | 0 | 0 | 30.58 |
| L | L | L | G | 11 | 0 | 26.51 |
| L | L | L | G | 0 | 0 | 26.51 |
| L | L | G | L | 11 | 0 | 31.33 |
| L | L | G | L | 0 | 0 | 31.32 |
| L | L | G | G | 11 | 0 | 29.25 |
| L | L | G | G | 0 | 0 | 29.25 |
| L | G | L | L | 11 | 0 | 29.55 |
| L | G | L | L | 0 | 0 | 29.55 |
| L | G | L | G | 11 | 0 | 27.48 |
| L | G | L | G | 0 | 0 | 27.48 |
| L | G | G | L | 11 | 0 | 30.30 |
| L | G | G | L | 0 | 0 | 30.30 |
| L | G | G | G | 11 | 0 | 30.22 |
| L | G | G | G | 0 | 0 | 30.22 |
| G | L | L | L | 11 | 0 | 29.56 |
| G | L | L | L | 0 | 0 | 29.56 |
| G | L | L | G | 11 | 0 | 25.49 |
| G | L | L | G | 0 | 0 | 25.49 |
| G | L | G | L | 11 | 0 | 32.29 |
| G | L | G | L | 0 | 0 | 32.29 |
| G | L | G | G | 11 | 0 | 30.22 |
| G | L | G | G | 0 | 0 | 30.22 |
| G | G | L | L | 11 | 0 | 28.53 |
| G | G | L | L | 0 | 0 | 28.53 |
| G | G | L | G | 11 | 0 | 26.46 |
| G | G | L | G | 0 | 0 | 26.46 |
| G | G | G | L | 11 | 0 | 31.26 |

| | | | | | | | | | | | | |
|---|--|---|--|---|--|---|--|----|--|---|--|-------|
| G | | G | | G | | L | | 0 | | 0 | | 31.26 |
| G | | G | | G | | G | | 11 | | 0 | | 31.18 |
| G | | G | | G | | G | | 0 | | 0 | | 31.18 |

6.91 FFT-4096

Функции

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

Прямое быстрое преобразование Фурье-4096.

6.91.1 Подробное описание

6.91.2 Функции

6.91.2.1 FFT_Fwd4096()

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

Прямое быстрое преобразование Фурье-4096.

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

Аргументы

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

Возвращает

void

Заметки

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

\perf

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | CLOCKS

| | | | | | | | | |
|---|--|---|--|---|--|---|--|-------|
| L | | L | | L | | L | | 38.25 |
| L | | L | | L | | G | | 26.82 |
| L | | L | | G | | L | | 32.21 |
| L | | L | | G | | G | | 30.74 |

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

6.92 IFFT-4096

Функции

- void `FFT_Inv4096` (`nm32sc *GSrcBuffer`, `nm32sc *GDstBuffer`, `void *LBuffer`, `void *GBuffer`)
Обратное быстрое преобразование Фурье. ОБПФ-4096.

6.92.1 Подробное описание

6.92.2 Функции

6.92.2.1 `FFT_Inv4096()`

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

Обратное быстрое преобразование Фурье. ОБПФ-4096.

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

Аргументы

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

Возвращает

void

Заметки

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

\perf

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | CLOCKS

| | | | | | | | | |
|---|--|---|--|---|--|---|--|-------|
| L | | L | | L | | L | | 38.25 |
| L | | L | | L | | G | | 26.82 |
| L | | L | | G | | L | | 32.21 |
| L | | L | | G | | G | | 30.74 |

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

6.93 FFT-8192

Функции

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

Прямое быстрое преобразование Фурье-8192.

6.93.1 Подробное описание

6.93.2 Функции

6.93.2.1 FFT_Fwd8192()

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

Прямое быстрое преобразование Фурье-8192.

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

Аргументы

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

Возвращает

void

Заметки

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

\perf

LSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

| | | | | |
|---|---|---|---|-------|
| L | L | L | L | 40.70 |
| L | L | L | G | 28.89 |
| L | L | G | L | 35.55 |
| L | L | G | G | 31.88 |
| L | G | L | L | 39.67 |

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

6.94 IFFT-8192

Функции

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

Обратное быстрое преобразование Фурье. ОБПФ-8192.

6.94.1 Подробное описание

6.94.2 Функции

6.94.2.1 `FFT_Inv8192()`

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

Обратное быстрое преобразование Фурье. ОБПФ-8192.

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

Аргументы

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

Возвращает

void

Заметки

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

\perf

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

| | | | | | | | | |
|---|--|---|--|---|--|---|--|-------|
| L | | L | | L | | L | | 40.70 |
| L | | L | | L | | G | | 28.89 |

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

6.95 Свертка

Группы

- [SIG_XCorr](#)

Свертка двух векторов.

6.95.1 Подробное описание

6.96 Масочная фильтрация

Группы

- [SIG_Median3](#)
Вычисление медианы трех чисел
- [КИХ-фильтрация](#)
Одномерная КИХ-фильтрация.

6.96.1 Подробное описание

6.97 Изменение размеров

Группы

- [SIG_ResampleDown2](#)
Уменьшение числа отсчетов в двое.
- [SIG_ResampleUp3Down2](#)
Передискретизации сигнала в $3/2$ раза
Передискретизации сигнала осуществляется методом Polyphase:
- [SIG_CreateResample](#)
Создание ядра для функции передискретизации SIG_Resample().
Функции выделяют память и инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.
- [SIG_SetResample](#)
Создание ядра для функции передискретизации SIG_Resample().
Функции инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.
- [SIG_Resample_perf](#)
Функции для оценки производительности функций фильтрации SIG_Resample()
Функция эквивалентно следующим псевдоинструкциям:

6.97.1 Подробное описание

6.98 Быстрое преобразование Фурье

Группы

- [DFT-8](#)

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

- [FFT-16](#)
- [FFT-32](#)
- [FFT-64](#)
- [FFT-128](#)
- [FFT-256](#)
- [FFT-512](#)
- [FFT-1024](#)
- [FFT-2048](#)
- [IDFT-8](#)
- [IFFT-16](#)
- [IFFT-32](#)
- [IFFT-64](#)
- [IFFT-128](#)
- [IFFT-256](#)
- [IFFT-512](#)
- [IFFT-1024](#)
- [IFFT-2048](#)
- [FFT-Common](#)
- [IFFT-Common](#)

Функции

- [int nmppsFFTFree_32fc \(NmppsFFTSpec_32fc *spec\)](#)

Функция освобождает память, выделенную под коэффициенты, необходимые для вычисления БПФ определенного размера

6.98.1 Подробное описание

6.98.2 Функции

6.98.2.1 nmppsFFTFree_32fc()

```
int nmppsFFTFree_32fc (  
    NmppsFFTSpec\_32fc * spec )
```

Функция освобождает память, выделенную под коэффициенты, необходимые для вычисления БПФ определенного размера

Аргументы

| | | |
|----|------|---|
| in | спес | структра, содержащая необходимые коэффиценты, для вычисления обратного БПФ определенного размера |
|----|------|---|

Возвращает

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

6.99 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)

6.99.1 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 [1,2,3,4...nSrcVecSize-1]. |
| nSrcVecSize | Размер входного вектора в элементах .Размер кратен 8,4 или 2 согласно типу данных. |

Возвращаемые значения

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

Заметки

По сути функции осуществляют фильтрацию данных окном свертки.

6.100 SIG_Median3

Вычисление медианы трех чисел

Функции

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

6.100.1 Подробное описание

Вычисление медианы трех чисел

Аргументы

| | |
|---|--------------|
| a | Первое число |
| b | Второе число |
| c | Третье число |

Возвращает

Медианное значение

6.101 КИХ-фильтрация

Одномерная КИХ-фильтрация.

Группы

- [nmppsFIR_Xs](#)
Одномерная КИХ-фильтрация
- [nmppsFIRInit_Xs](#)
Инициализация функции одномерной фильтрации
- [nmppsFIRInitAlloc_Xs](#)
Выделение и инициализация служебной структуры для функции одномерной фильтрации
- [nmppsFIRGetStateSize_Xs](#)
Возвращает размер памяти (в 32р.-словах) необходимый для хранения служебной структуры
- [nmppsFIRFree](#)
освобождает структуру pState в куче

6.101.1 Подробное описание

Одномерная КИХ-фильтрация.

6.102 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)

6.102.1 Подробное описание

Одномерная КИХ-фильтрация

Аргументы

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

Заметки

Инициализация служебной структуры производится соответствующей функцией nmppsFIR←Init_Xs() или nmppsFIRInitAlloc_Xs(). Максимальная производительность достигается при размещении pSrc, pDst и pPstate в разных банках памяти .

6.103 nmppsFIRInit_Xs

Инициализация функции одномерной фильтрации

Функции

- int nmppsFIRInit_8s (NmppsFIRState *pState, int *pTaps, int tapsLen)
- int nmppsFIRInit_8s16s (NmppsFIRState *pState, int *pTaps, int tapsLen)
- int nmppsFIRInit_8s32s (NmppsFIRState *pState, int *pTaps, int tapsLen)
- int nmppsFIRInit_16s (NmppsFIRState *pState, int *pTaps, int tapsLen)
- int nmppsFIRInit_16s32s (NmppsFIRState *pState, int *pTaps, int tapsLen)
- int nmppsFIRInit_32s (NmppsFIRState *pState, int *pTaps, int tapsLen)

6.103.1 Подробное описание

Инициализация функции одномерной фильтрации

Функция преобразует таблицу весовых коэффициентов окна фильтра во внутренний формат

Аргументы

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

Возвращает

Размер проинициализированной структуры pState в 32р. словах

6.104 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)

6.104.1 Подробное описание

Выделение и инициализация служебной структуры для функции одномерной фильтрации

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

Аргументы

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

Возвращаемые значения

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

Возвращает

Размер проинициализированной структуры pState в 32р. словах

6.105 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)

6.105.1 Подробное описание

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

Аргументы

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

Возвращает

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

6.106 nmppsFIRFree

освобождает структуру pState в куче

Функции

- void nmppsFIRFree (NmppsFIRState *pState)

6.106.1 Подробное описание

освобождает структуру pState в куче

Аргументы

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

6.107 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)

6.107.1 Подробное описание

Уменьшение числа отсчетов в двое.

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

Аргументы

| | |
|---------|--------------------------------|
| pSrcVec | Входной сигнал. |
| nSize | Размер массива входных данных. |

Возвращаемые значения

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

Заметки

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

6.108 SIG_ResampleUp3Down2

Передискретизации сигнала в 3/2 раза

Передискретизации сигнала осуществляется методом Polyphase:

.

Функции

- void SIG_ResampleUp3Down2 (nm8s *pSrcVec, nm16s *pDstVec, int nSrcVecSize, nm64s *pKernel)

6.108.1 Подробное описание

Передискретизации сигнала в 3/2 раза

Передискретизации сигнала осуществляется методом Polyphase:

.

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.109 SIG_CreateResample

Создание ядра для функции передискретизации SIG_Resample().

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

Функции

- void SIG_CreateResampleUp3Down2_8s16s (nm64s **pKernel, int nHint=MEM_LOCAL)
- void SIG_CreateResampleDown2_8u8u (nm64s **pKernel, int nHint=MEM_LOCAL)
- void SIG_CreateResampleDown2_16u16u (nm64s **pKernel, int nHint=MEM_LOCAL)

6.109.1 Подробное описание

Создание ядра для функции передискретизации SIG_Resample().

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

Аргументы

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

Возвращаемые значения

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

Заметки

Используется перед вызовом функции SIG_Filter.

6.110 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)

6.110.1 Подробное описание

Создание ядра для функции передискретизации SIG_Resample().

Функции инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.

Аргументы

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

Возвращает

Размер проинициализированной структуры pKernel в 32р. словах

Заметки

Используется перед вызовом функции SIG_Filter.

Используется перед вызовом функции SIG_Resample().

6.111 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)

6.111.1 Подробное описание

Функции для оценки производительности функций фильтрации SIG_Resample()

Функция эквивалентно следующим псевдоинструкциям:

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

\retval pKernel

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

Аргументы

| | |
|-------------|--------------------------------------|
| pSrcVec | Входной вектор. |
| nSrcVecSize | Размер входного вектора в элементах. |

Возвращаемые значения

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

Заметки

Инициализация служебной структуры производится соответствующей функцией SIG_Set↵Filter() и SIG_CreateFilter().

6.112 Типы векторных данных

Классы

- struct [v16nm4s](#)
- struct [v4nm8s](#)
- struct [s_v8nm8s](#)
- struct [s_v16nm8s](#)
- struct [s_v4nm16s](#)
- struct [s_v8nm16s](#)
- struct [s_v16nm16s](#)
- struct [s_v2nm32s](#)
- struct [s_v4nm32s](#)
- struct [s_v8nm32s](#)
- struct [s_v16nm32s](#)
- struct [s_v16nm4u](#)
- struct [s_v4nm8u](#)
- struct [s_v8nm8u](#)
- struct [s_v16nm8u](#)
- struct [s_v4nm16u](#)
- struct [s_v8nm16u](#)
- struct [s_v16nm16u](#)
- struct [s_v2nm32u](#)
- struct [s_v4nm32u](#)
- struct [s_v8nm32u](#)
- struct [s_v16nm32u](#)

Определения типов

- typedef void [nm1](#)
- typedef void [nm2s](#)
- typedef void [nm4s](#)
- typedef void [nm8s](#)
- typedef signed char [nm8s7b](#)
- typedef void [nm16s](#)
- typedef [nm16s](#) [nm16s15b](#)
- typedef int [nm32s](#)
- typedef int [nm32s31b](#)
- typedef int [nm32s30b](#)
- typedef long long [nm64s](#)
- typedef [nm64s](#) [nm64s63b](#)
- typedef void [nm2u](#)
- typedef void [nm4u](#)
- typedef [nm4u](#) [nm4u3b](#)
- typedef unsigned char [nm8u7b](#)
- typedef void [nm16u](#)
- typedef [nm16u](#) [nm16u15b](#)
- typedef unsigned int [nm32u](#)
- typedef unsigned int [nm32u31b](#)
- typedef unsigned long long [nm64u](#)
- typedef struct [s_v8nm8s](#) [v8nm8s](#)
- typedef struct [s_v16nm8s](#) [v16nm8s](#)
- typedef struct [s_v4nm16s](#) [v4nm16s](#)

- typedef struct [s_v8nm16s](#) [v8nm16s](#)
- typedef struct [s_v16nm16s](#) [v16nm16s](#)
- typedef struct [s_v2nm32s](#) [v2nm32s](#)
- typedef struct [s_v4nm32s](#) [v4nm32s](#)
- typedef struct [s_v8nm32s](#) [v8nm32s](#)
- typedef struct [s_v16nm32s](#) [v16nm32s](#)
- typedef [v16nm8s](#) [v16nm8s7b](#)
- typedef struct [s_v16nm4u](#) [v16nm4u](#)
- typedef struct [s_v4nm8u](#) [v4nm8u](#)
- typedef struct [s_v8nm8u](#) [v8nm8u](#)
- typedef struct [s_v16nm8u](#) [v16nm8u](#)
- typedef struct [s_v4nm16u](#) [v4nm16u](#)
- typedef struct [s_v8nm16u](#) [v8nm16u](#)
- typedef struct [s_v16nm16u](#) [v16nm16u](#)
- typedef struct [s_v2nm32u](#) [v2nm32u](#)
- typedef struct [s_v4nm32u](#) [v4nm32u](#)
- typedef struct [s_v8nm32u](#) [v8nm32u](#)
- typedef struct [s_v16nm32u](#) [v16nm32u](#)
- typedef [v16nm4u](#) [v16nm4b3u](#)

6.112.1 Подробное описание

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

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

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

Расшифровка мнемоники в названии типа:

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

6.112.2 Типы

6.112.2.1 `nm1`

```
typedef void nm1
```

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

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

Диапазон значений:

$[-1, 0]$

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

6.112.2.2 `nm16s`

```
typedef void nm16s
```

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

Диапазон значений:

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

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

6.112.2.3 nm16s15b

```
typedef nm16s nm16s15b
```

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

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

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 231

6.112.2.4 nm16u

```
typedef void nm16u
```

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

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

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 447

6.112.2.5 nm16u15b

```
typedef nm16u nm16u15b
```

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

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

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 463

6.112.2.6 nm2s

```
typedef void nm2s
```

Тип характеризует векторные данные как массив 2-х разрядных чисел со знаком.
Начальный адрес массива должен быть выровнен по границе 64-х 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

6.112.2.7 nm2u

```
typedef void nm2u
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 369

6.112.2.8 nm32s

```
typedef int nm32s
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 257

6.112.2.9 nm32s30b

```
typedef int nm32s30b
```

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

Диапазон значений:

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

См. определение в файле nmtypе.h строка 319

6.112.2.10 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:

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

См. определение в файле nmtypе.h строка 299

6.112.2.11 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

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

См. определение в файле nmtypе.h строка 477

6.112.2.12 nm32u31b

```
typedef unsigned int nm32u31b
```

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

Диапазон значений

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

См. определение в файле nmtype.h строка 491

6.112.2.13 nm4s

```
typedef void nm4s
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 126

6.112.2.14 nm4u

```
typedef void nm4u
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 382

6.112.2.15 nm4u3b

```
typedef nm4u nm4u3b
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 395

6.112.2.16 nm64s

```
typedef long long nm64s
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 340

6.112.2.17 nm64s63b

```
typedef nm64s nm64s63b
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 355

6.112.2.18 nm64u

```
typedef unsigned long long nm64u
```

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

Диапазон значений

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

См. определение в файле nmtype.h строка 504

6.112.2.19 nm8s

```
typedef void nm8s
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 149

6.112.2.20 nm8s7b

```
typedef signed char nm8s7b
```

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

Диапазон значений:

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

См. определение в файле nmtype.h строка 178

6.112.2.21 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 64-х слов. Предполагается, что размер массива данного типа как минимум кратен 8.

Диапазон значений:

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

См. определение в файле nmtypе.h строка 431

6.112.2.22 v16nm16s

```
typedef struct s_v16nm16s v16nm16s
```

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

6.112.2.23 v16nm16u

```
typedef struct s_v16nm16u v16nm16u
```

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

6.112.2.24 v16nm32s

```
typedef struct s_v16nm32s v16nm32s
```

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

6.112.2.25 v16nm32u

```
typedef struct s_v16nm32u v16nm32u
```

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

6.112.2.26 v16nm4b3u

```
typedef v16nm4u v16nm4b3u
```

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

Диапазон значений элементов структуры:

$$[0, \dots, 7]$$

См. определение в файле nmtypе.h строка 1225

6.112.2.27 v16nm4u

```
typedef struct s_v16nm4u v16nm4u
```

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

6.112.2.28 v16nm8s

```
typedef struct s_v16nm8s v16nm8s
```

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

6.112.2.29 v16nm8s7b

```
typedef v16nm8s v16nm8s7b
```

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

Диапазон значений элементов структуры:

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

См. определение в файле nmtype.h строка 1087

6.112.2.30 v16nm8u

```
typedef struct s_v16nm8u v16nm8u
```

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

6.112.2.31 v2nm32s

```
typedef struct s_v2nm32s v2nm32s
```

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

6.112.2.32 v2nm32u

```
typedef struct s_v2nm32u v2nm32u
```

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

6.112.2.33 v4nm16s

```
typedef struct s_v4nm16s v4nm16s
```

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

6.112.2.34 `v4nm16u`

```
typedef struct s_v4nm16u v4nm16u
```

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

6.112.2.35 `v4nm32s`

```
typedef struct s_v4nm32s v4nm32s
```

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

6.112.2.36 `v4nm32u`

```
typedef struct s_v4nm32u v4nm32u
```

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

6.112.2.37 `v4nm8u`

```
typedef struct s_v4nm8u v4nm8u
```

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

6.112.2.38 `v8nm16s`

```
typedef struct s_v8nm16s v8nm16s
```

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

6.112.2.39 `v8nm16u`

```
typedef struct s_v8nm16u v8nm16u
```

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

6.112.2.40 `v8nm32s`

```
typedef struct s_v8nm32s v8nm32s
```

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

6.112.2.41 `v8nm32u`

```
typedef struct s_v8nm32u v8nm32u
```

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

6.112.2.42 `v8nm8s`

```
typedef struct s_v8nm8s v8nm8s
```

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

6.112.2.43 `v8nm8u`

```
typedef struct s_v8nm8u v8nm8u
```

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

6.113 Типы скалярных данных

Определения типов

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

6.113.1 Подробное описание

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

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

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

6.113.2 Типы

6.113.2.1 int15b

```
typedef int int15b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 599

6.113.2.2 int16b

```
typedef int int16b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 612

6.113.2.3 int1b

```
typedef int int1b
```

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

Диапазон значений:

$$[-1, 0]$$

См. определение в файле nmtypes.h строка 521

6.113.2.4 int2b

```
typedef int int2b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 534

6.113.2.5 int30b

```
typedef int int30b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 625

6.113.2.6 int31b

```
typedef int int31b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 638

6.113.2.7 int32b

```
typedef int int32b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 651

6.113.2.8 int3b

```
typedef int int3b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 547

6.113.2.9 int4b

```
typedef int int4b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 560

6.113.2.10 int63b

```
typedef INT64 int63b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 664

6.113.2.11 int64b

```
typedef INT64 int64b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 677

6.113.2.12 int7b

```
typedef int int7b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 573

6.113.2.13 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 nmtypes.h строка 586

6.113.2.14 uint15b

```
typedef unsigned int uint15b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 768

6.113.2.15 uint16b

```
typedef unsigned int uint16b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 781

6.113.2.16 uint1b

```
typedef unsigned int uint1b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 690

6.113.2.17 uint2b

```
typedef unsigned int uint2b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 703

6.113.2.18 uint31b

```
typedef unsigned int uint31b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 794

6.113.2.19 uint32b

```
typedef unsigned int uint32b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 807

6.113.2.20 uint3b

```
typedef unsigned int uint3b
```

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

Диапазон значений:

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

См. определение в файле nmtypes.h строка 716

6.113.2.21 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 729

6.113.2.22 uint63b

```
typedef UINT64 uint63b
```

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

Диапазон значений:

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

См. определение в файле `nmtype.h` строка 820

6.113.2.23 uint64b

```
typedef nm64u uint64b
```

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

Диапазон значений:

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

См. определение в файле `nmtype.h` строка 833

6.113.2.24 uint7b

```
typedef unsigned int uint7b
```

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

Диапазон значений:

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

См. определение в файле `nmtype.h` строка 742

6.113.2.25 uint8b

```
typedef unsigned int uint8b
```

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

Диапазон значений:

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

См. определение в файле `nmtype.h` строка 755

6.114 Функции поддержки

Группы

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

6.114.1 Подробное описание

6.115 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.

6.115.1 Подробное описание

6.116 Арифметические операции

Группы

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

6.116.1 Подробное описание

6.117 Логические и бинарные операции

Группы

- [nmppsNot_](#)
Функция логического "НЕ".
- [nmppsAndC](#)
Функция логического "И" между вектором и константой.
- [nmppsAnd](#)
Функция логического "И" между двумя векторами.
- [nmppsAnd4V_](#)
Функция логического "И" между четырьмя векторами.
- [nmppsAndNotV_](#)
Функция логического "И-НЕ" между двумя векторами.
- [nmppsOrC](#)
Функция логического "ИЛИ" между вектором и константой.
- [nmppsOr](#)
Функция логического "ИЛИ" между двумя векторами.
- [nmppsOr3V_](#)
Функция логического "ИЛИ" между четырьмя векторами.
- [nmppsOr4V_](#)
Функция логического "ИЛИ" между четырьмя векторами.
- [nmppsXorC](#)
Функция логического "Исключающего ИЛИ" между вектором и константой.
- [nmppsXor](#)
Функция логического "Исключающего ИЛИ" между двумя векторами.
- [nmppsMaskV_](#)
Функция логического ИЛИ с предварительным маскированием двух векторов.
- [nmppsRShiftC](#)
Операция арифметического сдвига вправо.
- [nmppsRShiftC_](#)
Операция логического сдвига.
- [nmppsRShiftC_AddC_](#)
Операция логического сдвига.
- [nmppsDisplaceBits](#)
Непрерывное смещение битов внутри бинарного массива в сторону конца массива
Функция смещает биты внутри бинарного массива на несколько позиций (nBits) в сторону конца массива.
Внутри 64р. слова младшие биты сдвигаются на старшие позиции того же слова, а старшие биты перемещаются в младшие позиции следующего 64р. слова. Освободившееся место в первом 64р. слове заполняется старшими битами 64р. слова с адреса pnBits. Сдвинутые биты сохраняются в массиве pDst.
Пример сдвига на 8 бит :

6.117.1 Подробное описание

6.118 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.

6.118.1 Подробное описание

6.119 Переупорядочивание и сортировка

Группы

- [VEC_QSort](#)
Сортировка массива по убыванию.
- [nmppsRemap_](#)
Переупорядочивание элементов вектора по таблице.
- [nmppSplit](#)
Расщепляет массив на два, группируя по четным и нечетным элементам
- [nmppSplitEco](#)
Расщепляет массив на два, группируя по четным и нечетным элементам

6.119.1 Подробное описание

6.120 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)

6.120.1 Подробное описание

Вычисление абсолютных значений для элементов вектора.

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

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

Restrictions:

Значения элементов вектора не должны быть равны минимальному значению для соответствующего типа (т.е. -128, -2¹⁵ и т.д). В противном случае, абсолютное значение для таких элементов вычисляется неверно, давая на выходе то же самое число.

6.121 nmppsAbs1

Функция логического вычисления модулей элементов вектора.

Функции

- void nmppsAbs1_4s (const nm4s *pSrcVec, nm4s *pDstVec, int nSize)
- void nmppsAbs1_8s (const nm8s *pSrcVec, nm8s *pDstVec, int nSize)
- void nmppsAbs1_16s (const nm16s *pSrcVec, nm16s *pDstVec, int nSize)
- void nmppsAbs1_32s (const nm32s *pSrcVec, nm32s *pDstVec, int nSize)
- void nmppsAbs1_64s (const nm64s *pSrcVec, nm64s *pDstVec, int nSize)

6.121.1 Подробное описание

Функция логического вычисления модулей элементов вектора.

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

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.122 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)

6.122.1 Подробное описание

Изменение знака элементов вектора на противоположный.

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

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.123 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)

6.123.1 Подробное описание

Добавление к вектору константы.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.124 nmppsAdd

Сложение двух векторов.

Функции

- void nmppsAdd_4s (const nm4s *pSrcVec1, const nm4s *pSrcVec2, nm4s *pDstVec, int nSize)
- void nmppsAdd_8s (const nm8s *pSrcVec1, const nm8s *pSrcVec2, nm8s *pDstVec, int nSize)
- void nmppsAdd_16s (const nm16s *pSrcVec1, const nm16s *pSrcVec2, nm16s *pDstVec, int nSize)
- void nmppsAdd_32s (const nm32s *pSrcVec1, const nm32s *pSrcVec2, nm32s *pDstVec, int nSize)
- void nmppsAdd_64s (const nm64s *pSrcVec1, const nm64s *pSrcVec2, nm64s *pDstVec, int nSize)
- void nmppsAdd_32f (const nm32f *pSrcVec1, const nm32f *pSrcVec2, nm32f *pDstVec, int nSize)
- void nmppsAddEx_64s (const nm64s *pSrcVec1, int srcStep1, const nm64s *pSrcVec2, int srcStep2, nm64s *pDstVec, int dstStep, int nSize)

6.124.1 Подробное описание

Сложение двух векторов.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.125 nmppsAdd_AddC

Сложение двух векторов с прибавлением константы.

Функции

- void nmppsAdd_AddC_32s (nm32s *pSrcVec1, nm32s *pSrcVec2, int nVal, nm32s *pDstVec, int nSize)

6.125.1 Подробное описание

Сложение двух векторов с прибавлением константы.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.126 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)

6.126.1 Подробное описание

Вычитание константы из вектора.

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

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

Аргументы

| | |
|---------|------------------------------------|
| pSrcVec | Входной вектор. |
| nVal | Вычитаемая константа. |
| pnVal | Указатель на вычитаемую константу. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.127 nmppsSubCRev

Вычитание константы из вектора с переменной знака элементов вектора.

Функции

- void nmppsSubCRev_8s (const nm8s *pSrcVec, int8b nVal, nm8s *pDstVec, int nSize)
- void nmppsSubCRev_16s (const nm16s *pSrcVec, int16b nVal, nm16s *pDstVec, int nSize)
- void nmppsSubCRev_32s (const nm32s *pSrcVec, int32b nVal, nm32s *pDstVec, int nSize)
- void nmppsSubCRev_64s (const nm64s *pSrcVec, int64b nVal, nm64s *pDstVec, int nSize)

6.127.1 Подробное описание

Вычитание константы из вектора с переменной знака элементов вектора.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.128 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)

6.128.1 Подробное описание

Вычитание двух вектров.

$$pDstVec[i] = pSrcVec1[i] - pSrcVec2[i]$$

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

Аргументы

| | |
|----------|------------------------------|
| pSrcVec1 | Уменьшаемый вектор. |
| pSrcVec2 | Вычитаемый вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.129 nmppsAbsDiff

Вычисление вектора модулей разности элементов двух векторов.

Функции

- void nmppsAbsDiff_8s (const nm8s *pSrcVec1, nm8s *pSrcVec2, nm8s *pDstVec, int nSize)
- void nmppsAbsDiff_16s (const nm16s *pSrcVec1, nm16s *pSrcVec2, nm16s *pDstVec, int nSize)
- void nmppsAbsDiff_32s (const nm32s *pSrcVec1, nm32s *pSrcVec2, nm32s *pDstVec, int nSize)
- void nmppsAbsDiff_64s (const nm64s *pSrcVec1, nm64s *pSrcVec2, nm64s *pDstVec, int nSize)

6.129.1 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, -2¹⁵ и т.д.). В противном случае, абсолютное значение для таких элементов вычисляется не верно, давая на выходе то же самое число.

6.130 nmppsAbsDiff1

Функция логического вычисления модулей разностей элементов двух векторов.

Функции

- void nmppsAbsDiff1_8s (nm8s *pSrcVec1, nm8s *pSrcVec2, nm8s *pDstVec, int nSize)

6.130.1 Подробное описание

Функция логического вычисления модулей разностей элементов двух векторов.

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

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

Аргументы

| | |
|----------|------------------------------|
| pSrcVec1 | Входной вектор. |
| pSrcVec2 | Вычитаемый вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

Restrictions:

Разность элементов векторов не должна быть равна минимальному значению для соответствующего типа (т.е. -128, -2¹⁵ и т.д.). В противном случае, абсолютное значение для таких элементов вычисляется не верно, давая на выходе то же самое число.

6.131 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)

6.131.1 Подробное описание

Умножение вектора на константу.

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

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nVal | Константа-множитель. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.132 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)

6.132.1 Подробное описание

Поэлементное умножение векторов с прибавлением константы.

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

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

Аргументы

| | |
|----------|----------------------------------|
| pSrcVec1 | Входной вектор. |
| pSrcVec2 | Входной вектор. |
| nVal | указаель на константу-инкремент. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.133 nmppsMulC_AddC

Умножение вектора на константу с прибавлением константы.

Функции

- void nmppsMulC_AddC_32s (const nm32s *pSrcVec, int nMulVal, int nAddVal, nm32s *pDstVec, int nSize)
- void nmppsMulC_AddC_2x32s (int32x2 *dataSparseSrc, int32x2 *mulArg, int32x2 *addArg, int32x2 *dataSparseDst, int size, int stepSparseSrc, int stepSparseDst)

Sparse vector by constant multiplication with addition of constant.

$$dataSparseDst[i \cdot stepSparseDst][k] = dataSparseSrc[i \cdot stepSparseSrc][k] \cdot mulArg[k] + addArg[k],$$

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

where K is value of intWxK type.

6.133.1 Подробное описание

Умножение вектора на константу с прибавлением константы.

$$pDstVec[i] = nMulVal \cdot pSrcVec[i] + nAddVal,$$

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nMulVal | Константа-множитель. |
| nAddVal | Добавляемая константа. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.133.2 Функции

6.133.2.1 nmppsMulC_AddC_2x32s()

```
void nmppsMulC_AddC_2x32s (
    int32x2 * dataSparseSrc,
    int32x2 * mulArg,
    int32x2 * addArg,
    int32x2 * dataSparseDst,
    int size,
    int stepSparseSrc,
    int stepSparseDst )
```

Sparse vector by constant multiplication with addition of constant.

$$dataSparseDst[i \cdot stepSparseDst][k] = dataSparseSrc[i \cdot stepSparseSrc][k] \cdot mulArg[k] + addArg[k],$$

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

where K is value of int WxK type.

Аргументы

| | | |
|----|---------------|--|
| in | dataSparseSrc | Input sparse vector of 64-bit packed words |
| in | mulArg | Packed 64-bit word with values to multiply |
| in | addArg | Packed 64-bit word with values to add |
| in | dataSparseDst | Ouput sparse vector of 64-bit packed words |
| in | size | actual amount of 64-bit packed words in sparse vector to be processed |
| in | stepSparseSrc | 64-bit step between input packed words in memory . By default=1 means that input vector is continuous |
| in | stepSparseDst | 64-bit step between output packed words in memory. By default=1 means that output vector is continuous |

Возвращает

void

6.134 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)

6.134.1 Подробное описание

Sparse vector by constant multiplication with addition of constant.

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

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

where K is value of int WxK type

Аргументы

| | | |
|----|---------------|--|
| in | dataSparseSrc | Input sparse vector of 64-bit packed words |
| in | preshiftArg | Packed 64-bit word of values for preshifting of input data = [2,4,6,8...28,30] |
| in | mulArg | Packed 64-bit word with values to multiply |
| in | addArg | Packed 64-bit word with values to add |
| in | dataSparseDst | Ouput sparse vector of 64-bit packed words |
| in | size | actual amount of 64-bit packed words in sparse vector to be processed |
| in | stepSparseSrc | 64-bit step between input packed words in memory . By default=1 means that input vector is continuous |
| in | stepSparseDst | 64-bit step between output packed words in memory. By default=1 means that output vector is continuous |

Возвращает

void

6.135 nmppsMulC_AddV_AddC

Умножение вектора на константу с прибавлением вектора и константы.

Функции

- void nmppsMulC_AddV_AddC_32s (nm32s *pSrcVec1, int nMulVal, nm32s *pSrcVec2, int nAddVal, nm32s *pDstVec, int nSize)

6.135.1 Подробное описание

Умножение вектора на константу с прибавлением вектора и константы.

$$pDstVec[i] = nMulVal \cdot pSrcVec1[i] + pSrcVec2[i] + nAddVal,$$

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

Аргументы

| | |
|----------|------------------------------|
| pSrcVec1 | Первый входной вектор. |
| nMulVal | Константа-множитель. |
| pSrcVec2 | Второй входной вектор. |
| nAddVal | Добавляемая константа. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.136 nmppsSumN

Сложение нескольких векторов.

Функции

- void nmppsSumN_8s16s (nm8s **ppSrcVec, nm16s *pDstVec, int nSize, int nNumberOfVectors)
- void nmppsSumN_16s (nm16s **ppSrcVec, nm16s *pDstVec, int nSize, int nNumberOfVectors)

6.136.1 Подробное описание

Сложение нескольких векторов.

$$pDstVec[i] = \sum_{j=0}^{(nNumberOfVectors-1)} ppSrcVec(j)(i)$$

Аргументы

| | |
|------------------|---|
| ppSrcVec | Массив указателей на суммируемые вектора. |
| nNumberOfVectors | Число суммируемых векторов. |
| nSize | Размер векторов в элементах = [32*PACK] |

Возвращаемые значения

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

Возвращает

void

6.137 nmppsDivC

Деление вектора на константу.

Функции

- void nmppsDivC_32s (nm32s *pSrcVec, int nDivisor, nm32s *pDstVec, int nSize, void *pTmpBuf1, void *pTmpBuf2)

6.137.1 Подробное описание

Деление вектора на константу.

$$pDstVec[i] = \frac{pSrcVec[i]}{Divisor},$$

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

Аргументы

| | |
|----------|--|
| pSrcVec | Входной вектор. |
| nDivisor | Константа-делитель. |
| nSize | Размер входного вектора в элементах. |
| pTmpBuf1 | Временный массив размером nSize 64-х разрядных слов. |
| pTmpBuf2 | Временный массив размером nSize 64-х разрядных слов. |

Возвращаемые значения

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

Возвращает

void

Restrictions:

- Допустимые значения для элементов входного вектора лежат в диапазоне [-4095,...,4095];
- Допустимые значения для делителя лежат в диапазоне [1,2,..145].

Заметки

Для корректного использования in-place параметров следует учитывать порядок получения промежуточных результатов:

the order of obtaining intermediate results:

pSrcVec => pTmpBuf1 (1cpl:L<=>G) - 1/x Multiplying (in-place is supported)

pTmpBuf1 => pTmpBuf2 (1cpl:G<=>L) - Scaling down (in-place is supported)

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

Примеры использования in-place параметров:

```
nmppsDiv_(L0,G0,10240,3,G0,L0);
```

```
nmppsDiv_(L0,L0,10240,3,L0,L0);
```

6.138 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)

6.138.1 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.

6.139 nmppsDotProd

Скалярное умножение двух векторов.

Функции

- `int nmppsDotProd_8s8sm (const nm8s *pSrcVec1, const nm8s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `int nmppsDotProd_8s16sm (const nm8s *pSrcVec1, const nm16s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `int nmppsDotProd_8s32sm (const nm8s *pSrcVec1, const nm32s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `void nmppsDotProd_8s64s (const nm8s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`
- `int nmppsDotProd_16s16sm (const nm16s *pSrcVec1, const nm16s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `int nmppsDotProd_16s32sm (const nm16s *pSrcVec1, const nm32s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `void nmppsDotProd_16s64s (const nm16s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`
- `int nmppsDotProd_32s32sm (const nm32s *pSrcVec1, const nm32s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `void nmppsDotProd_32s64s (const nm32s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`
- `void nmppsDotProd_64s64s (const nm64s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`

6.139.1 Подробное описание

Скалярное умножение двух векторов.

$$nRes = \sum_{i=0}^{nSize-1} pSrcVec1[i] \cdot pSrcVec2[i]$$

Аргументы

| | |
|----------|------------------------------|
| pSrcVec1 | Первый вектор. |
| pSrcVec2 | Второй вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

| | |
|-------|---------------------------------------|
| pnRes | Указатель на результирующее значение. |
|-------|---------------------------------------|

Возвращает

pTmpBuff Временный массив из nSize элементов.
void

6.140 nmppsWeightedSum

Поэлементное взвешенное суммирование элементов двух векторов

Функции

- void nmppsWeightedSum_8s16s (nm8s *pSrcVec1, int nW1, nm8s *pSrcVec2, int nW2, nm16s *pDstVec, int nSize)
- void nmppsWeightedSum_16s32s (nm16s *pSrcVec1, int nW1, nm16s *pSrcVec2, int nW2, nm32s *pDstVec, int nSize)
- void nmppsWeightedSum_32s64s (nm32s *pSrcVec1, nm64s nW1, nm32s *pSrcVec2, nm64s nW2, nm64s *pDstVec, int nSize)

6.140.1 Подробное описание

Поэлементное взвешенное суммирование элементов двух векторов

$$pDstVec[i] = nW1 \cdot pSrcVec1[i] + nW2 \cdot pSrcVec2[i],$$

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

\param pSrcVec1

1-ый входной вектор.

\param nW1

1-ый весовой коэффициент

\param pSrcVec2

2-ой входной вектор.

\param nW2

2-ой весовой коэффициент

\param nSize

Размер векторов в элементах.

\retval pDstVec

Результирующий вектор.

Возвращает

void

6.141 nmppsNot_

Функция логического "НЕ".

Функции

- void nmppsNot_2u (const nm2u *pSrcVec, nm2u *pDstVec, int nSize)
- void nmppsNot_8u (const nm8u *pSrcVec, nm8u *pDstVec, int nSize)
- void nmppsNot_16u (const nm16u *pSrcVec, nm16u *pDstVec, int nSize)
- void nmppsNot_32u (const nm32u *pSrcVec, nm32u *pDstVec, int nSize)
- void nmppsNot_64u (const nm64u *pSrcVec, nm64u *pDstVec, int nSize)

6.141.1 Подробное описание

Функция логического "НЕ".

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

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

Функция изменяет значения всех битов входного вектора на противоположные.

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.142 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)

6.142.1 Подробное описание

Функция логического "И" между вектором и константой.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.143 nmppsAnd

Функция логического "И" между двумя векторами.

Функции

- void nmppsAnd_1 (const nm1 *pSrcVec1, const nm1 *pSrcVec2, nm1 *pDstVec, int nSize)
- void nmppsAnd_2u (const nm2u *pSrcVec1, const nm2u *pSrcVec2, nm2u *pDstVec, int nSize)
- void nmppsAnd_4u (const nm4u *pSrcVec1, const nm4u *pSrcVec2, nm4u *pDstVec, int nSize)
- void nmppsAnd_8u (const nm8u *pSrcVec1, const nm8u *pSrcVec2, nm8u *pDstVec, int nSize)
- void nmppsAnd_16u (const nm16u *pSrcVec1, const nm16u *pSrcVec2, nm16u *pDstVec, int nSize)
- void nmppsAnd_32u (const nm32u *pSrcVec1, const nm32u *pSrcVec2, nm32u *pDstVec, int nSize)
- void nmppsAnd_64u (const nm64u *pSrcVec1, const nm64u *pSrcVec2, nm64u *pDstVec, int nSize)

6.143.1 Подробное описание

Функция логического "И" между двумя векторами.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.144 nmppsAnd4V_

Функция логического "И" между четырьмя векторами.

Функции

- void nmppsAnd4V_64u (nm64u *pSrcVec1, nm64u *pSrcVec2, nm64u *pSrcVec3, nm64u *pSrcVec4, nm64u *pDstVec, int nSize)

6.144.1 Подробное описание

Функция логического "И" между четырьмя векторами.

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

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

Аргументы

| | |
|----------|------------------------------|
| pSrcVec1 | Первый входной вектор. |
| pSrcVec2 | Второй входной вектор. |
| pSrcVec3 | Третий входной вектор. |
| pSrcVec4 | Четвертый входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.145 nmppsAndNotV_

Функция логического "И-НЕ" между двумя векторами.

Функции

- void nmppsAndNotV_64u (nm64u *pSrcVec1, nm64u *pSrcVec2, nm64u *pDstVec, int nSize)

6.145.1 Подробное описание

Функция логического "И-НЕ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \wedge not pSrcVec2[i]$$

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.146 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)

6.146.1 Подробное описание

Функция логического "ИЛИ" между вектором и константой.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.147 nmppsOr

Функция логического "ИЛИ" между двумя векторами.

Функции

- void nmppsOr_1 (const nm1 *pSrcVec1, const nm1 *pSrcVec2, nm1 *pDstVec, int nSize)
- void nmppsOr_2u (const nm2u *pSrcVec1, const nm2u *pSrcVec2, nm2u *pDstVec, int nSize)
- void nmppsOr_4u (const nm4u *pSrcVec1, const nm4u *pSrcVec2, nm4u *pDstVec, int nSize)
- void nmppsOr_8u (const nm8u *pSrcVec1, const nm8u *pSrcVec2, nm8u *pDstVec, int nSize)
- void nmppsOr_16u (const nm16u *pSrcVec1, const nm16u *pSrcVec2, nm16u *pDstVec, int nSize)
- void nmppsOr_32u (const nm32u *pSrcVec1, const nm32u *pSrcVec2, nm32u *pDstVec, int nSize)
- void nmppsOr_64u (const nm64u *pSrcVec1, const nm64u *pSrcVec2, nm64u *pDstVec, int nSize)

6.147.1 Подробное описание

Функция логического "ИЛИ" между двумя векторами.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.148 nmppsOr3V_

Функция логического "ИЛИ" между четырьмя векторами.

Функции

- void nmppsOr3V_64u (nm64u *pSrcVec1, nm64u *pSrcVec2, nm64u *pSrcVec3, nm64u *pDstVec, int nSize)

6.148.1 Подробное описание

Функция логического "ИЛИ" между четырьмя векторами.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.149 nmppsOr4V_

Функция логического "ИЛИ" между четырьмя векторами.

Функции

- void nmppsOr4V_64u (nm64u *pSrcVec1, nm64u *pSrcVec2, nm64u *pSrcVec3, nm64u *pSrcVec4, nm64u *pDstVec, int nSize)

6.149.1 Подробное описание

Функция логического "ИЛИ" между четырьмя векторами.

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

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

Аргументы

| | |
|----------|------------------------------|
| pSrcVec1 | Первый входной вектор. |
| pSrcVec2 | Второй входной вектор. |
| pSrcVec3 | Третий входной вектор. |
| pSrcVec4 | Четвертый входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.150 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)

6.150.1 Подробное описание

Функция логического "Исключающего ИЛИ" между вектором и константой.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.151 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)

6.151.1 Подробное описание

Функция логического "Исключающего ИЛИ" между двумя векторами.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.152 nmppsMaskV_

Функция логического ИЛИ с предварительным маскированием двух векторов.

Функции

- void nmppsMaskV_64u (nm64u *pSrcVec1, nm64u *pSrcVec2, nm64u *pMaskVec, nm64u *pDstVec, int nSize)

6.152.1 Подробное описание

Функция логического ИЛИ с предварительным маскированием двух векторов.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.153 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)

6.153.1 Подробное описание

Операция арифметического сдвига вправо.

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

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

Функции реализуют операции арифметического сдвига вправо битов элементов вектора. Освободившиеся биты заполняются знаковым битом - старшим битом.

Аргументы

| | |
|---------|---|
| pSrcVec | Входной вектор. |
| nSize | Размер вектора в элементах. |
| nShift | Параметр определяет на сколько позиций нужно сдвинуть биты элементов вектора. |

Возвращаемые значения

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

Возвращает

void

6.154 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)

6.154.1 Подробное описание

Операция логического сдвига.

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

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

Функции реализуют операции логического сдвига вправо битов элементов вектора. Сдвиг осуществляется на число бит, указанных в соответствующем операнде. Освободившиеся биты заполняются нулями.

Аргументы

| | |
|---------|---|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |
| nShift | Определяет на сколько позиций нужно сдвинуть биты элемента вектора. |

Возвращаемые значения

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

Возвращает

void

6.155 nmppsRShiftC_AddC_

Операция логического сдвига.

Операция логического сдвига.

$$pDstVec[i] = (pSrcVec[i] >> nShift) + nAddVal,$$

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

Функции реализуют операции логического сдвига вправо битов элементов вектора с прибавлением константы. Сдвиг осуществляется на число бит, указанных в соответствующем операнде. Освободившиеся биты заполняются нулями.

Аргументы

| | |
|---------|---|
| pSrcVec | Входной вектор. |
| nAddVal | Константа для суммирования. |
| nSize | Размер векторов в элементах. |
| nShift | Определяет на сколько позиций нужно сдвинуть биты элемента вектора. |

Возвращаемые значения

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

Возвращает

void

6.156 nmppsDisplaceBits

Непрерывное смещение битов внутри бинарного массива в сторону конца массива

Функция смещает биты внутри бинарного массива на несколько позиций (nBits) в сторону конца массива. Внутри 64р. слова младшие биты сдвигаются на старшие позиции того же слова, а старшие биты перемещаются в младшие позиции следующего 64р. слова. Освободившееся место в первом 64р. слове заполняется старшими битами 64р. слова с адреса pnBits. Сдвинутые биты сохраняются в массиве pDst. Пример сдвига на 8 бит :

.

Функции

- void nmppsFwdShiftBitstream (const nm64u *pSrcVec, nm64u *pDstVec, nm64u *pnBits, int n← Bits, int nSize)

6.156.1 Подробное описание

Непрерывное смещение битов внутри бинарного массива в сторону конца массива

Функция смещает биты внутри бинарного массива на несколько позиций (nBits) в сторону конца массива. Внутри 64р. слова младшие биты сдвигаются на старшие позиции того же слова, а старшие биты перемещаются в младшие позиции следующего 64р. слова. Освободившееся место в первом 64р. слове заполняется старшими битами 64р. слова с адреса pnBits. Сдвинутые биты сохраняются в массиве pDst. Пример сдвига на 8 бит :

.

```
pnBits=[AB00000000000000]
pSrcVec=[0807060504030201][FF0F0E0D0C0B0A09]
pDstVec=[07060504030201AB][0F0E0D0C0B0A0908]
```

Последние 8 бит массива pDstVec будут потеряны. Если же указатель pBits установить на последнее 64р. слово в результате получится цикличическое перемещение бит.

Аргументы

| | |
|---------|---|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в 64р. элементах. |
| pnBits | Указатель на 64р-слово, старшие биты которого записываются на освобождающуюся при сдвиге младшую часть первого 64р. слова |
| nBits | Кол-во позиций на которое происходит смещение бит :nBits=[2,4,6....62]. |

Возвращаемые значения

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

Возвращает

void

6.157 nmppsSet-инициализация

Функция инициализации элементов массива постоянным значением.

Функции

- void nmppsSet_8s (nm8s *pVec, int8b nVal, int nSize)
- void nmppsSet_16s (nm16s *pVec, int16b nVal, int nSize)
- void nmppsSet_32s (nm32s *pVec, int32b nVal, int nSize)
- void nmppsSet_64sp (nm64s *pVec, int64b *nVal, int nSize)
- __INLINE__ void nmppsSet_64s (nm64s *pVec, int64b nVal, int nSize)
- __INLINE__ void nmppsSet_8u (nm8u *pVec, uint8b nVal, int nSize)
- __INLINE__ void nmppsSet_16u (nm16u *pVec, uint16b nVal, int nSize)
- __INLINE__ void nmppsSet_32u (nm32u *pVec, uint32b nVal, int nSize)
- __INLINE__ void nmppsSet_64u (nm64u *pVec, uint64b nVal, int nSize)
- __INLINE__ void nmppsSet_64up (nm64u *pVec, uint64b *nVal, int nSize)

6.157.1 Подробное описание

Функция инициализации элементов массива постоянным значением.

$$pVec[i] = nVal,$$

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

Аргументы

| | |
|-------|--|
| nSize | Размер вектора в элементах. |
| nVal | Константа. Диапазон значений nVal должен соответствовать типу результирующего вектора. |

Возвращаемые значения

| | |
|------|------------------------|
| pVec | Результирующий вектор. |
|------|------------------------|

Возвращает

void

6.158 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)

6.158.1 Подробное описание

Инициализация массива случайными числами.

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.158.2 Функции

6.158.2.1 nmppsRandUniform_64s()

```
void nmppsRandUniform_64s (
    nm64s * pDstVec,
    int nSize )
```

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

6.159 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 ()`

6.159.1 Подробное описание

Генерация случайного числа с равномерным распределением.

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

Аргументы

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

Возвращает

`int` Случайное число в диапазоне либо `[nMin, nMax]`. Для функции без параметров данный диапазон `[-231; 231-1]`.

6.160 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)

6.160.1 Подробное описание

Инициализация массива элементами арифметической прогрессии.

$$pVec[i] = nOffset + nSlope \cdot i$$

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

Аргументы

| | |
|---------|--|
| nOffset | Первый член арифметической прогрессии. |
| nSlope | Разность арифметической прогрессии. |
| nSize | Размер вектора. |

Возвращаемые значения

| | |
|------|------------------------|
| pVec | Результирующий массив. |
|------|------------------------|

Возвращает

void

6.161 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_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 nmppsConvertRisc_8u32u (const nm8u *pSrcVec, nm32u *pDstVec, int nSize)

Функция конвертации вектора беззнаковых байт в вектор беззнаковых целых чисел

6.161.1 Подробное описание

Изменение разрядности элементов вектора.

Преобразование знаковых данных к меньшей разрядности осуществляется отбрасыванием старших битов. Преобразование знаковых данных к большей разрядности осуществляется с распространением влево старшего (знакового) бита. Преобразование беззнаковых данных к большей разрядности осуществляется добавлением слева старших нулевых битов.

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.161.2 Функции

6.161.2.1 nmppsConvert_1s2s()

```
void nmppsConvert_1s2s (
    const nm1 * pSrcVec,
    nm2s * pDstVec,
    int nSize )
```

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

6.161.2.2 nmppsConvert_1u2u()

```
void nmppsConvert_1u2u (
    const nm1 * pSrcVec,
    nm2u * pDstVec,
    int nSize )
```

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

6.161.2.3 nmppsConvert_32s32fcr()

```
void nmppsConvert_32s32fcr (
    const nm32s * pSrcVec,
    nm32fcr * pDstVec,
    int nSize )
```

Функция конвертации вектора целых чисел в вектор комплексных чисел, где мнимая(равна 0) и действительная части - 32-битные числа с плавающей точкой

Аргументы

| | |
|---------|---|
| pSrcVec | указатель на входной вектор целых чисел |
| pDstVec | указатель на выходной вектор комплексных чисел с плавающей точкой |
| nSize | число элементов во входном векторе |

После конвертации мнимая часть каждого комплексного числа в выходном векторе будет равна 0

Функция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

6.161.2.4 nmppsConvert_32sc32fcr()

```
void nmppsConvert_32sc32fcr (
    const nm32sc * pSrcVec,
    nm32fcr * pDstVec,
    int nSize )
```

Функция конвертации вектора комплексных чисел с целыми действительной и мнимой частью (32 бита) в вектор комплексных чисел, где мнимая и действительная части - 32-битные числа с плавающей точкой

Аргументы

| | |
|---------|---|
| pSrcVec | указатель на входной вектор комплексных чисел с плавающей точкой |
| pDstVec | указатель на выходной вектор комплексных чисел с плавающей точкой |
| nSize | число элементов во входном векторе |

Функция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

6.161.2.5 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 |

Функция выполняется на RISC-процессоре

6.162 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)

6.162.1 Подробное описание

Копирование вектора.

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

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.163 nmppsCoryua_

Копирование вектора с невыровненной байтовой позиции в выровненную.

Функции

- void nmppsCoryua_8s (const nm8s *pSrcVec, int nSrcOffset, nm8s *pDstVec, int nSize)

6.163.1 Подробное описание

Копирование вектора с невыровненной байтовой позиции в выровненную.

$$pDstVec[i] = pSrcVec[nSrcOffset + i]$$

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

Позиция байта считается выровненной если она совпадает с границей 64р. слов в памяти.

\param pSrcVec

Входной вектор.

Аргументы

| | |
|------------|--|
| pDstVec | Результирующий вектор. |
| nSrcOffset | Смещение в элементах относительно начала вектора. nSrcOffset Может принимать любое значение. |
| nSize | Кол-во копируемых элементов. |

Возвращает

void

6.164 nmppsSwap_

Перестановка двух векторов.

Функции

- void nmppsSwap_64s (nm64s *pSrcVec1, nm64s *pSrcVec2, int nSize)

6.164.1 Подробное описание

Перестановка двух векторов.

Аргументы

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

Возвращает

void

6.165 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)

6.165.1 Подробное описание

Поиск значения максимального элемента вектора.

$$nMaxValue = \max_i (pSrcVec[i])$$

Аргументы

| | |
|---------|-----------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер вектора в элементах. |

Возвращаемые значения

| | |
|-----------|---|
| nMaxValue | значение максимального элемент вектора. |
|-----------|---|

Возвращает

void

Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

6.165.2 Функции

6.165.2.1 nmppsMax_16s15b()

```
void nmppsMax_16s15b (
    const nm16s15b * pSrcVec,
    int nSize,
    int16b * nMaxValue )
```

Restrictions:

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

6.165.2.2 nmppsMax_32s31b()

```
void nmppsMax_32s31b (
    const nm32s31b * pSrcVec,
    int nSize,
    int * nMaxValue )
```

Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на $2^{31}-1$.
Примеры допустимых диапазонов входных чисел:
[00000000h..7FFFFFFFh]=[0.. $2^{31}-1$] [FFFFFFFFh..7FFFFFFEh]=[-1.. $2^{31}-2$] [C0000000h..3←
FFFFFFFh]=[- 2^{30} .. $2^{30}-1$] [80000000h..00000000h]=[- 2^{31} ..0]

6.165.2.3 nmppsMax_8s7b()

```
void nmppsMax_8s7b (
    const nm8s7b * pSrcVec,
    int nSize,
    int8b * nMaxValue )
```

Restrictions:

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

Максимальный и минимальный элементы массива должны отличаться не более чем на 127.
Примеры допустимых диапазонов входных чисел:
[00h..7Fh]=[0.. 2^7-1] [FFh..7Eh]=[-1.. 2^7-1] [C0h..3Fh]=[- 2^6 .. 2^6-1] [80h..00h]=[- 2^7 ..0]

6.166 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)

6.166.1 Подробное описание

Поиск значения минимального элемента вектора.

$$nMinValue = \min_i (pSrcVec[i])$$

Аргументы

| | |
|---------|-----------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер вектора в элементах. |

Возвращаемые значения

| | |
|-----------|--------------------------------|
| nMinValue | значение минимального элемент. |
|-----------|--------------------------------|

Возвращает

void

Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

6.166.2 Функции

6.166.2.1 nmppsMin_16s15b()

```
void nmppsMin_16s15b (
    const nm16s15b * pSrcVec,
    int nSize,
    int16b * nMinValue )
```

Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на $2^{15}-1$.
 Примеры допустимых диапазонов входных чисел:
 $[0000h..7FFFh] = [0..+32767]$ $[FFFFh..7FFEh] = [-1..+32766]$ $[C000h..3FFFh] = [-16384..+16383]$ $[8000h..0000h] = [-32768..0]$

6.166.2.2 nmppsMin_32s31b()

```
void nmppsMin_32s31b (
    const nm32s31b * pSrcVec,
    int nSize,
    int * nMinValue )
```

Restrictions:

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

6.166.2.3 nmppsMin_8s7b()

```
void nmppsMin_8s7b (
    const nm8s7b * pSrcVec,
    int nSize,
    int8b * nMinValue )
```

Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на 127.
 Примеры допустимых диапазонов входных чисел:
 $[00h..7Fh] = [0..+127]$ $[FFh..7Eh] = [-1..+126]$ $[C0h..3Fh] = [-64..+63]$ $[80h..00h] = [-128..0]$

6.167 nmppsMaxIndx_

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

Функции

- void nmppsMaxIndx_8s (nm8s7b *pSrcVec, int nSize, int *nIndex, int8b *nMaxValue, void *pLTmpBuf, void *pGTmpBuf, int nSearchDir)
- void nmppsMaxIndx_16s (nm16s15b *pSrcVec, int nSize, int *nIndex, int16b *nMaxValue, void *pLTmpBuf, void *pGTmpBuf, int nSearchDir)
- void nmppsMaxIndx_32s (nm32s31b *pSrcVec, int nSize, int *nIndex, int32b *nMaxValue, void *pLTmpBuf, void *pGTmpBuf, int nSearchDir)

6.167.1 Подробное описание

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

$$nMaxValue = \max_i (pSrcVec[i])$$

Аргументы

| | |
|------------|---|
| pSrcVec | Входной вектор. |
| nSize | Размер вектора в элементах. Занимаемый этим вектором объем памяти должен быть кратен 64 длинным словам (nm64s[64,128,...]). |
| pLTmpBuf | Временный массив на локальной шине из nSize элементов. |
| pGTmpBuf | Временный массив на глобальной шине . |
| nSearchDir | Направление поиска максимума. При nSearchDir = 1, поиск ведется от начала массива. При nSearchDir = -1, поиск ведется от конца массива. Значение максимального элемента. |

Возвращаемые значения

| | |
|--------|---|
| nIndex | Индекс первого найденного максимума среди равных. |
|--------|---|

Возвращает

void

Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

Restrictions:

Диапазоны входных элементов могут быть "плавающими"
Например для данных nm16s15b максимальный и минимальный элементы массива должны отличаться не более чем на $2^{15}-1$. Примеры допустимых диапазонов входных чисел для типа nm16s15b :

Here are some examples of admissible ranges for input numbers: \n

[0000h..7FFFh]=[0..+32767] [FFFFh..7FFEh]=[-1..+32766] [C000h..3FFFh]=[-16384..+16383]
[8000h..0000h]=[-32768..0]

6.168 nmppsMinIndx_

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

Функции

- void nmppsMinIndx_8s (nm8s7b *pSrcVec, int nSize, int *nIndex, int8b *nMinValue, void *pLTmpBuf, void *pGTmpBuf, int nSearchDir)
- void nmppsMinIndx_16s (nm16s15b *pSrcVec, int nSize, int *nIndex, int16b *nMinValue, void *pLTmpBuf, void *pGTmpBuf, int nSearchDir)
- void nmppsMinIndx_32s (nm32s31b *pSrcVec, int nSize, int *nIndex, int32b *nMinValue, void *pLTmpBuf, void *pGTmpBuf, int nSearchDir)

6.168.1 Подробное описание

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

$$nMinValue = \min_i (pSrcVec[i])$$

Аргументы

| | |
|------------|---|
| pSrcVec | Входной вектор. |
| nSize | Размер вектора в элементах. Занимаемый этим вектором объем памяти должен быть кратен 64 длинным словам (nm64s[64,128,...]). |
| pLTmpBuf | Временный массив на локальной шине . |
| pGTmpBuf | Временный массив на глобальной шине из nSize элементов. |
| nSearchDir | Направление поиска минимума. При nSearchDir = 1, поиск ведется от начала массива. При nSearchDir = -1, поиск ведется от конца массива. Значение минимального элемента. |

Возвращаемые значения

| | |
|--------|--|
| nIndex | Индекс первого найденного минимума среди равных. |
|--------|--|

Возвращает

void

Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

Restrictions:

Диапазоны входных элементов могут быть "плавающими"
Например для данных nm16s15b максимальный и минимальный элементы массива должны отличаться не более чем на $2^{15}-1$. Примеры допустимых диапазонов входных чисел для типа nm16s15b :

Here are some examples of admissible ranges for input numbers: \n

[0000h..7FFFh]=[0..+32767] [FFFFh..7FFEh]=[-1..+32766] [C000h..3FFFh]=[-16384..+16383]
[8000h..0000h]=[-32768..0]

6.169 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)

6.169.1 Подробное описание

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

Аргументы

| | |
|---------|---|
| pSrcVec | Массив из N элементов, где N определяется числом в имени функции. |
| nStride | шаг между элементами в 32р. словах |

Возвращаемые значения

| | |
|--------|------------------------------|
| Индекс | первого найденного минимума. |
|--------|------------------------------|

Возвращает

Значение минимального элемента.

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные pSrcVec может ссылаться на нечетный адрес.

6.170 nmppsFirstZeroIndx

Поиск позиции первого нулевого элемента в векторе .

Функции

- `int nmppsFirstZeroIndx_32s (int *pSrcVec, int nSize)`

6.170.1 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.

6.171 nmppsFirstNonZeroIndx

Поиск позиции первого ненулевого элемента в векторе .

Функции

- `int nmppsFirstNonZeroIndx_32s (int *pSrcVec, int nSize)`

6.171.1 Подробное описание

Поиск позиции первого ненулевого элемента в векторе .

Аргументы

| | |
|----------------------|-----------------|
| <code>pSrcVec</code> | Входной массив. |
| <code>nIndex</code> | Размер массива. |

Возвращает

Позиция ненулевого элемента или -1 в случае если ненулевой элемент не найден

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится до первого ненулевого элемента.

6.172 nmppsLastZeroIndx

Поиск позиции последнего нулевого элемента в векторе .

Функции

- `int nmppsLastZeroIndx_32s (int *pSrcVec, int nSize)`

6.172.1 Подробное описание

Поиск позиции последнего нулевого элемента в векторе .

Аргументы

| | |
|----------------------|-----------------|
| <code>pSrcVec</code> | Входной массив. |
| <code>nIndex</code> | Размер массива. |

Возвращает

Позиция нулевого элемента или -1 в случае если нулевой элемент не найден

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится с конца до первого нулевого элемента.

6.173 nmppsLastNonZeroIndx

Поиск позиции последнего ненулевого элемента в векторе .

Функции

- `int nmppsLastNonZeroIndx_32s (int *pSrcVec, int nSize)`

6.173.1 Подробное описание

Поиск позиции последнего ненулевого элемента в векторе .

Аргументы

| | |
|----------------------|-----------------|
| <code>pSrcVec</code> | Входной массив. |
| <code>nIndex</code> | Размер массива. |

Возвращает

Позиция нулевого элемента или -1 в случае если нулевой элемент не найден

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится с конца до первого ненулевого элемента.

6.174 nmppsMinEvery_

Поэлементный минимум из двух векторов.

Функции

- void nmppsMinEvery_8s (nm8s7b *pSrcVec1, nm8s7b *pSrcVec2, nm8s7b *pDstMinVec, int nSize)
- void nmppsMinEvery_16s (nm16s15b *pSrcVec1, nm16s15b *pSrcVec2, nm16s15b *pDstMinVec, int nSize)
- void nmppsMinEvery_32s (nm32s31b *pSrcVec1, nm32s31b *pSrcVec2, nm32s31b *pDstMinVec, int nSize)
- void nmppsMinEvery_64s (nm64s63b *pSrcVec1, nm64s63b *pSrcVec2, nm64s63b *pDstMinVec, int nSize)

6.174.1 Подробное описание

Поэлементный минимум из двух векторов.

$$pDstMinVec[i] = \min(pSrcVec1[i], pSrcVec2[i])$$

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

Аргументы

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

Возвращаемые значения

| | |
|------------|---|
| pDstMinVec | Массив поэлементных минимумов для входных массивов. |
|------------|---|

Возвращает

void

Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на $2^{15} - 1$.

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than $2^{15} - 1$.

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[0..+32767]

[FFFFh..7FFEh]=[-1..+32766]

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

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

6.175 nmppsMaxEvery_

Поэлементный максимум из двух векторов.

Функции

- void nmppsMaxEvery_8s (nm8s7b *pSrcVec1, nm8s7b *pSrcVec2, nm8s7b *pDstMaxVec, int nSize)
- void nmppsMaxEvery_16s (nm16s15b *pSrcVec1, nm16s15b *pSrcVec2, nm16s15b *pDstMaxVec, int nSize)
- void nmppsMaxEvery_32s (nm32s31b *pSrcVec1, nm32s31b *pSrcVec2, nm32s31b *pDstMaxVec, int nSize)
- void nmppsMaxEvery_64s (nm64s63b *pSrcVec1, nm64s63b *pSrcVec2, nm64s63b *pDstMaxVec, int nSize)

6.175.1 Подробное описание

Поэлементный максимум из двух векторов.

$$pDstMaxVec[i] = \max(pSrcVec1[i], pSrcVec2[i])$$

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

Аргументы

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

Возвращаемые значения

| | |
|------------|---|
| pDstMaxVec | Массив поэлементных минимумов для входных массивов. |
|------------|---|

Возвращает

void

Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на $2^{15} - 1$.

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than $2^{15} - 1$.

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[0..+32767]

[FFFFh..7FFEh]=[-1..+32766]

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

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

6.176 nmppsMinCmpLtV_

Поэлементный минимум из двух векторов.

Функции

- void nmppsMinCmpLtV_16s (nm16s15b *pSrcVec1, nm16s15b *pSrcVec2, nm16s15b *pDstMin, nm16s15b *pDstSignMask, int nSize)

6.176.1 Подробное описание

Поэлементный минимум из двух векторов.

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

$$pDstSignMask[i] = \begin{cases} 11 \dots 1b, & pSrcVec1[i] < pSrcVec2[i] \\ 00 \dots 0b, & pSrcVec1[i] \leq pSrcVec2[i] \end{cases}$$

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

Аргументы

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

Возвращаемые значения

| | |
|--------------|--|
| pDstMin | Массив поэлементных минимумов для входных массивов. |
| pDstSignMask | Массив знаков поэлементных разностей первого и второго векторов. |

Возвращает

void

Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на $2^{15} - 1$.

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than $2^{15} - 1$.

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[0..+32767]

[FFFFh..7FFEh]=[-1..+32766]

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

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

6.177 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)

6.177.1 Подробное описание

Сравнивает элементы массива на меньше нуля.

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

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

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

Аргументы

| | |
|---------|------------------------------|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.178 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)

6.178.1 Подробное описание

Сравнивает элементы массива на признак равенства нулю.

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

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

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

Аргументы

| | |
|-----------|--|
| pSrcVec | Входной вектор. |
| nSize | Размер векторов в элементах. Vector size in elements. |
| nTrueFlag | Младший бит nTrueFlag определяет значение выходного бита при выполнении условия. |

Возвращаемые значения

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

Возвращает

void

6.178.2 Функции

6.178.2.1 nmppsCmpEq0_16u15b()

```
void nmppsCmpEq0_16u15b (
    nm16u15b * pSrcVec,
    nm1 * pDstVec,
    int nSize,
    int nTrueFlag )
```

Заметки

nSize =[1,2,3,4...]

6.178.2.2 nmppsCmpEq0_32u31b()

```
void nmppsCmpEq0_32u31b (
    nm32u31b * pSrcVec,
    nm1 * pDstVec,
    int nSize,
    int nTrueFlag )
```

Заметки

nSize =[1,2,3,4...]

6.178.2.3 nmppsCmpEq0_8u7b()

```
void nmppsCmpEq0_8u7b (
    nm8u7b * pSrcVec,
    nm1 * pDstVec,
    int nSize,
    int nTrueFlag )
```

Заметки

nSize =[1,2,3,4...]

6.179 nmppsCmpMinMaxV_

Поэлементное сравнение двух векторов.

Функции

- void nmppsCmpMinMaxV_8s (nm8s *pSrcVec1, nm8s *pSrcVec2, nm8s *pDstMin, nm8s *pDstMax, int nSize)
- void nmppsCmpMinMaxV_16s (nm16s *pSrcVec1, nm16s *pSrcVec2, nm16s *pDstMin, nm16s *pDstMax, int nSize)
- void nmppsCmpMinMaxV_32s (nm32s *pSrcVec1, nm32s *pSrcVec2, nm32s *pDstMin, nm32s *pDstMax, int nSize)

6.179.1 Подробное описание

Поэлементное сравнение двух векторов.

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

$$pDstMax[i] = \max(pSrcVec1[i], pSrcVec2[i])$$

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

Аргументы

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

Возвращаемые значения

| | |
|---------|--|
| pDstMin | Массив поэлементных минимумов для входных массивов. |
| pDstMax | Массив поэлементных максимумов для входных массивов. |

Возвращает

void

Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на $2^{15} - 1$.

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than $2^{15} - 1$.

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[0..+32767]

[FFFFh..7FFEh]=[-1..+32766]
[C000h..3FFFh]=[-16384..+16383]
[8000h..0000h]=[-32768..0]

6.180 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)

6.180.1 Подробное описание

Функция насыщения.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.181 nmppsClipCC_

Функция насыщения с произвольными порогами.

Функции

- void nmppsClipCC_32s (nm32s30b *pSrcVec, int30b nNegThresh, int30b nPosThresh, nm32s30b *pDstVec, int nSize)

6.181.1 Подробное описание

Функция насыщения с произвольными порогами.

$$pDstVec[i] = \begin{cases} nPosThresh, & \text{if } pSrcVec[i] > nPosThresh \\ pSrcVec[i], & \text{if } nNegThresh \leq pSrcVec[i] \leq nPosThresh \\ nNegThresh, & \text{if } pSrcVec[i] < nNegThresh \end{cases}$$

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

Аргументы

| | |
|------------|-----------------------------|
| pSrcVec | Входной вектор. |
| nNegThresh | Нижний порог насыщения. |
| nPosThresh | Верхний порог насыщения. |
| nSize | Размер вектора в элементах. |

Возвращаемые значения

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

Возвращает

void

Restrictions:

- Диапазон изменения для \form#65.
- Диапазон изменения для \form#66. \n

6.182 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)

6.182.1 Подробное описание

Сокращение разрядности данных с предварительной их обработкой.

$$pDstVec[i] = Convert(Clip(pSrcVec[i], nClipFactor) >> nShift) + nAddValue$$

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

Сокращение разрядности данных выполняется после предварительной обработки и осуществляется путем отбрасывания старших битов.

Аргументы

| | |
|-------------|---|
| pSrcVec | Входной вектор. |
| nClipFactor | Показатель степени, определяющий верхний и нижний пороги насыщения. =[1,2,3...15] |
| nShift | Параметр определяет на сколько позиций =[2,4,6...14] нужно сдвинуть биты элементов вектора; |
| nAddValue | Добавляемая константа. |
| nSize | Размер векторов в элементах. |

Возвращаемые значения

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

Возвращает

void

6.183 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)

6.183.1 Подробное описание

Сокращение разрядности данных с предварительной их обработкой.

$$pDstVec[i] = Convert(Clip(pSrcVec[i], nClipFactor)) + nAddValue$$

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

Сокращение разрядности данных выполняется после предварительной обработки и осуществляется путем отбрасывания старших битов.

\param pSrcVec

Входной вектор.

Аргументы

| | |
|-------------|---|
| nClipFactor | Показатель степени, определяющий верхний и нижний пороги насыщения. = [1,2,3...15] |
| nAddValue | Добавляемая константа. |
| nSize | Размер векторов в элементах. указатель на матрицу коэффициентов для векторного умножителя. Для ускорения. |

Возвращаемые значения

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

Возвращает

void

6.184 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)

6.184.1 Подробное описание

Сравнивает элементы массива на признак равенства константе.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.185 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)

6.185.1 Подробное описание

Сравнивает элементы массива на признак неравенства нулю.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.186 nmppsCmpNeC

Сравнивает элементы массива на признак неравенства константе.

Группы

- [nmppsCmpEqV_](#)
Поэлементное сравнение элементов двух векторов на признак равенства.
- [nmppsCmpNeV_](#)
Поэлементное сравнение элементов двух векторов на признак неравенства.

Функции

- void nmppsCmpNeC_8s (const [nm8s](#) *pSrcVec, [int8b](#) nCmpVal, [nm8s](#) *pDstVec, int nSize)
- void nmppsCmpNeC_16s (const [nm16s](#) *pSrcVec, [int16b](#) nCmpVal, [nm16s](#) *pDstVec, int nSize)
- void nmppsCmpNeC_32s (const [nm32s](#) *pSrcVec, [int32b](#) nCmpVal, [nm32s](#) *pDstVec, int nSize)
- void nmppsCmpNeC_64s (const [nm64s](#) *pSrcVec, [int64b](#) nCmpVal, [nm64s](#) *pDstVec, int nSize)
- int nmppsCmpNeC_8s8um (const [nm8s](#) *pSrcVec, [int8b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- int nmppsCmpNeC_16s8um (const [nm16s](#) *pSrcVec, [int16b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- int nmppsCmpNeC_32s8um (const [nm32s](#) *pSrcVec, [int32b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- int nmppsCmpNeC_64s8um (const [nm64s](#) *pSrcVec, [int64b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- void nmppsCmpNeC_8u7b ([nm8u7b](#) *pSrcVec, [uint7b](#) nCmpVal, [nm8s](#) *pDstVec, int nSize, [int8b](#) nTrueFlag)
- void nmppsCmpNeC_16u15b ([nm16u15b](#) *pSrcVec, [uint15b](#) nCmpVal, [nm16s](#) *pDstVec, int nSize, [int16b](#) nTrueFlag)
- int nmppsCmpLtC_8s8um (const [nm8s](#) *pSrcVec, [int8b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- int nmppsCmpLtC_16s8um (const [nm16s](#) *pSrcVec, [int16b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- int nmppsCmpLtC_32s8um (const [nm32s](#) *pSrcVec, [int32b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- int nmppsCmpLtC_64s8um (const [nm64s](#) *pSrcVec, [int64b](#) nCmpVal, nm8u *pDstVec, int nSize, struct [NmppsTmpSpec](#) *spec)
- void nmppsCmpLtC_8s7b (const [nm8s7b](#) *pSrcVec, [int8b](#) nCmpVal, [nm8s](#) *pDstVec, int nSize)
- void nmppsCmpLtC_16s15b (const [nm16s15b](#) *pSrcVec, [int16b](#) nCmpVal, [nm16s](#) *pDstVec, int nSize)
- void nmppsCmpLtC_32s31b (const [nm32s31b](#) *pSrcVec, [int32b](#) nCmpVal, [nm32s](#) *pDstVec, int nSize)
- void nmppsCmpLtC_64s63b (const [nm64s63b](#) *pSrcVec, [int64b](#) nCmpVal, [nm64s](#) *pDstVec, int nSize)
- void nmppsCmpGtC_8s7b (const [nm8s7b](#) *pSrcVec, [int8b](#) nCmpVal, [nm8s](#) *pDstVec, int nSize)
- void nmppsCmpGtC_16s15b (const [nm16s15b](#) *pSrcVec, [int16b](#) nCmpVal, [nm16s](#) *pDstVec, int nSize)
- void nmppsCmpGtC_32s31b (const [nm32s31b](#) *pSrcVec, [int32b](#) nCmpVal, [nm32s](#) *pDstVec, int nSize)
- void nmppsCmpGtC_64s63b (const [nm64s63b](#) *pSrcVec, [int64b](#) nCmpVal, [nm64s](#) *pDstVec, int nSize)
- void nmppsCmpNe_8s (const [nm8s](#) *pSrcVec1, const [nm8s](#) *pSrcVec2, [nm8s](#) *pDstVec, int nSize)

- void nmppsCmpNe_16s (const nm16s *pSrcVec1, const nm16s *pSrcVec2, nm16s *pDstVec, int nSize)
- void nmppsCmpNe_32s (const nm32s *pSrcVec1, const nm32s *pSrcVec2, nm32s *pDstVec, int nSize)
- void nmppsCmpNe_64s (const nm64s *pSrcVec1, const nm64s *pSrcVec2, nm64s *pDstVec, int nSize)
- int nmppsCmpNe_8s8um (const nm8s *pSrcVec1, const nm8s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- int nmppsCmpNe_16s8um (const nm16s *pSrcVec1, const nm16s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- int nmppsCmpNe_32s8um (const nm32s *pSrcVec1, const nm32s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- int nmppsCmpNe_64s8um (const nm64s *pSrcVec1, const nm64s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- void nmppsCmpLt_8s7b (const nm8s *pSrcVec1, const nm8s *pSrcVec2, nm8s *pDstVec, int nSize)
- void nmppsCmpLt_16s15b (const nm16s *pSrcVec1, const nm16s *pSrcVec2, nm16s *pDstVec, int nSize)
- void nmppsCmpLt_32s31b (const nm32s *pSrcVec1, const nm32s *pSrcVec2, nm32s *pDstVec, int nSize)
- void nmppsCmpLt_64s63b (const nm64s *pSrcVec1, const nm64s *pSrcVec2, nm64s *pDstVec, int nSize)
- int nmppsCmpLt_8s8um (const nm8s *pSrcVec1, const nm8s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- int nmppsCmpLt_16s8um (const nm16s *pSrcVec1, const nm16s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- int nmppsCmpLt_32s8um (const nm32s *pSrcVec1, const nm32s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- int nmppsCmpLt_64s8um (const nm64s *pSrcVec1, const nm64s *pSrcVec2, nm8u *pDstVec, int nSize, struct NmppsTmpSpec *spec)
- void nmppsCmpNeV_8s8u (nm8s *src1, nm8s *src2, nm8u *dst, int nSize, int8b nTrueFlag)
- void Vec_ClipRShiftConvert_AddC (nm32s *pSrcVec, nm8u *pDstVec, int nSize)

6.186.1 Подробное описание

Сравнивает элементы массива на признак неравенства константе.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.187 nmppsCmpEqV_

Поэлементное сравнение элементов двух векторов на признак равенства.

Функции

- void nmppsCmpEqV_16u15b (nm16u15b *pSrcVec1, nm16u15b *pSrcVec2, nm16s *pDstVec, int nSize, int16b nTrueFlag)
- void nmppsCmpEqV_8u7b (nm8u7b *pSrcVec1, nm8u7b *pSrcVec2, nm8s *pDstVec, int nSize, int8b nTrueFlag)

6.187.1 Подробное описание

Поэлементное сравнение элементов двух векторов на признак равенства.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.188 nmppsCmpNeV_

Поэлементное сравнение элементов двух векторов на признак неравенства.

Функции

- void nmppsCmpNeV_16u (nm16u15b *pSrcVec1, nm16u15b *pSrcVec2, nm16s *pDstVec, int nSize, int16b nTrueFlag)
- void nmppsCmpNeV_8u (nm8u7b *pSrcVec1, nm8u7b *pSrcVec2, nm8s *pDstVec, int nSize, int8b nTrueFlag)

6.188.1 Подробное описание

Поэлементное сравнение элементов двух векторов на признак неравенства.

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

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

Аргументы

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

Возвращаемые значения

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

Возвращает

void

6.189 nmppsAddr_

Возвращает адрес ячейки памяти, содержащей указанный элемент.

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

Функции

- `__INLINE__ nm1 * nmppsAddr_1 (const nm1 *pVec, int nIndex)`
- `__INLINE__ nm2s * nmppsAddr_2s (const nm2s *pVec, int nIndex)`
- `__INLINE__ nm4s * nmppsAddr_4s (const nm4s *pVec, int nIndex)`
- `__INLINE__ nm8s * nmppsAddr_8s (const nm8s *pVec, int nIndex)`
- `__INLINE__ nm16s * nmppsAddr_16s (const nm16s *pVec, int nIndex)`
- `__INLINE__ nm32s * nmppsAddr_32s (const nm32s *pVec, int nIndex)`
- `__INLINE__ nm64s * nmppsAddr_64s (const nm64s *pVec, int nIndex)`
- `__INLINE__ nm2u * nmppsAddr_2u (const nm2u *pVec, int nIndex)`
- `__INLINE__ nm4u * nmppsAddr_4u (const nm4u *pVec, int nIndex)`
- `__INLINE__ nm8u * nmppsAddr_8u (const nm8u *pVec, int nIndex)`
- `__INLINE__ nm16u * nmppsAddr_16u (const nm16u *pVec, int nIndex)`
- `__INLINE__ nm32u * nmppsAddr_32u (const nm32u *pVec, int nIndex)`
- `__INLINE__ nm64u * nmppsAddr_64u (const nm64u *pVec, int nIndex)`

6.189.1 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_`.

6.190 nmppsSetVal_

Модификация элемента вектора.

Функции

- void nmppsPut_1 (nm1 *pVec, int nIndex, int1b nVal)
- void nmppsPut_2s (nm2s *pVec, int nIndex, int2b nVal)
- void nmppsPut_4s (nm4s *pVec, int nIndex, int4b nVal)
- void nmppsPut_8s (nm8s *pVec, int nIndex, int8b nVal)
- void nmppsPut_16s (nm16s *pVec, int nIndex, int16b nVal)
- __INLINE__ void nmppsPut_32s (nm32s *pVec, int nIndex, int32b nVal)
- __INLINE__ void nmppsPut_64s (nm64s *pVec, int nIndex, int64b nVal)
- __INLINE__ void nmppsPut_2u (nm2u *pVec, int nIndex, uint2b nVal)
- __INLINE__ void nmppsPut_4u (nm4u *pVec, int nIndex, uint4b nVal)
- __INLINE__ void nmppsPut_8u (nm8u *pVec, int nIndex, uint8b nVal)
- __INLINE__ void nmppsPut_16u (nm16u *pVec, int nIndex, uint16b nVal)
- __INLINE__ void nmppsPut_32u (nm32u *pVec, int nIndex, uint32b nVal)
- __INLINE__ void nmppsPut_64u (nm64u *pVec, int nIndex, uint64b nVal)

6.190.1 Подробное описание

Модификация элемента вектора.

$$pVec(nIndex) = Val$$

Аргументы

| | |
|--------|-------------------|
| pVec | Вектор. |
| nIndex | Позиция элемента |
| nVal | Значение элемента |

Возвращает

void

6.191 nmppsGetVal_

Извлекает значение элемента вектора.

Функции

- void nmppsGetVal_1 (nm1 *pVec, int nIndex, int1b *nVal)
- void nmppsGetVal_2s (nm2s *pVec, int nIndex, int2b *nVal)
- void nmppsGetVal_4s (nm4s *pVec, int nIndex, int4b *nVal)
- void nmppsGetVal_8s (nm8s *pVec, int nIndex, int8b *nVal)
- void nmppsGetVal_16s (nm16s *pVec, int nIndex, int16b *nVal)
- __INLINE__ void nmppsGetVal_32s (nm32s *pVec, int nIndex, int32b *nVal)
- __INLINE__ void nmppsGetVal_64s (nm64s *pVec, int nIndex, int64b *nVal)
- void nmppsGetVal_2u (nm2u *pVec, int nIndex, uint2b *nVal)
- void nmppsGetVal_4u (nm4u *pVec, int nIndex, uint4b *nVal)
- void nmppsGetVal_8u (nm8u *pVec, int nIndex, uint8b *nVal)
- void nmppsGetVal_16u (nm16u *pVec, int nIndex, uint16b *nVal)
- __INLINE__ void nmppsGetVal_32u (nm32u *pVec, int nIndex, uint32b *nVal)
- __INLINE__ void nmppsGetVal_64u (nm64u *pVec, int nIndex, uint64b *nVal)

6.191.1 Подробное описание

Извлекает значение элемента вектора.

Аргументы

| | |
|--------|-------------------|
| pVec | Вектор. |
| nIndex | Позиция элемента. |

Возвращаемые значения

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

Возвращает

void

6.192 nmppsGetVal_(return)

Извлекает значение элемента вектора.

Функции

- [int2b](#) nmppsGet_2s ([nm2s](#) *pVec, int nIndex)
- [int4b](#) nmppsGet_4s ([nm4s](#) *pVec, int nIndex)
- [int8b](#) nmppsGet_8s ([nm8s](#) *pVec, int nIndex)
- [int16b](#) nmppsGet_16s ([nm16s](#) *pVec, int nIndex)
- `__INLINE__` [int32b](#) nmppsGet_32s ([nm32s](#) *pVec, int nIndex)
- [uint1b](#) nmppsGet_1 ([nm1](#) *pVec, int nIndex)
- [uint2b](#) nmppsGet_2u ([nm2u](#) *pVec, int nIndex)
- [uint4b](#) nmppsGet_4u ([nm4u](#) *pVec, int nIndex)
- [uint8b](#) nmppsGet_8u ([nm8u](#) *pVec, int nIndex)
- [uint16b](#) nmppsGet_16u ([nm16u](#) *pVec, int nIndex)
- `__INLINE__` [uint32b](#) nmppsGet_32u ([nm32u](#) *pVec, int nIndex)

6.192.1 Подробное описание

Извлекает значение элемента вектора.

Аргументы

| | |
|--------|-------------------|
| pVec | Вектор. |
| nIndex | Позиция элемента. |

Возвращает

Значение элемента.

6.193 VEC_QSort

Сортировка массива по убыванию.

Функции

- void nmppsQSort_32s (nm32s *pSrcDstVec, int nSize)

6.193.1 Подробное описание

Сортировка массива по убыванию.

Аргументы

| | |
|------------|----------------------------------|
| pSrcDstVec | Входной и результирующий вектор. |
| nSize | Размер вектора в элементах. |

Возвращает

void

Restrictions:

Функция работает рекурсивно, передавая параметры через стек, поэтому размер стека должен быть больше $4 \cdot \log_2(nSize)$ 32-битных слов в лучшем случае (элементы массива расположены беспорядочно) и больше $6 \cdot nSize$ 32-битных слов в худшем случае (элементы массива уже упорядочены)

6.194 nmppsRemap_

Переупорядочивание элементов вектора по таблице.

Функции

- void nmppsRemap_32u (nm32u *pSrcVec, nm32u *pDstVec, nm32s *pRemapTable, int nDstVecSize)
- void nmppsRemap_8u (nm8u *pSrcVec, nm8u *pDstVec, nm32s *pRemapTable, int nSrcVecSize, int nDstVecSize, void *pTmpBuf1, void *pTmpBuf2)

6.194.1 Подробное описание

Переупорядочивание элементов вектора по таблице.

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

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

Аргументы

| | |
|-------------|--|
| pSrcVec | Входной вектор. |
| pRemapTable | Таблица новых индексов для переупорядочивания. |
| nDstVecSize | Размер результирующего вектора в элементах. |
| pTmpBuf1 | Временный массив nm32u pTmpBuf1[nSrcVecSize]. |
| pTmpBuf2 | Временный массив nm32u pTmpBuf2[nDstVecSize]. Результирующий вектор nm8u pDstVec[nDstVecSize]. |

Возвращает

void

```
// Функция
// void nmppsRemap_8u(nm8u* pSrcVec, nm8u* pDstVec, nm32s* pRemapTable, int nSrcVecSize, int nDstVecSize,
// void* pTmpBuf1, void* pTmpBuf2);
// выполняет следующие действия:
nmppsConvert_8u((nm8u*) pSrcVec, (nm32u*)pTmpBuf1,nSrcVecSize);
nmppsRemap_32u((nm32u*)pTmpBuf1,(nm32u*)pTmpBuf2,RemapTable,DstVecSize);
nmppsConvert_32s((nm32s*)pTmpBuf2,(nm8s*) pDstVec, DstVecSize);
```

Заметки

Возможность использования inplace параметров определяется исходя из последовательности процессов чтения/записи:

the folowing sequence of actions :

- pSrcVec => pTmpBuf1 - inplace запрещен;

- $pTmpBuf1 \Rightarrow pTmpBuf2$ - inplace запрещен;
- $pTmpBuf2 \Rightarrow pDstVec$ - inplace разрешен;

6.195 nmppSplit

Расщепляет массив на два, группируя по четным и нечетным элементам

Расщепляет массив на два, группируя по четным и нечетным элементам

Аргументы

| | | |
|-----|--------------|--|
| in | src | Входной массив |
| out | dst1 | Выходной массив размера size/2 |
| out | dst2 | Выходной массив размера size/2 |
| in | size | Размер исходного массива в элементах. Кратность параметра size должна соответствовать двум длинным 64-р. словам. |
| in | tmpSizeofDst | Временный массив размера size/2 |

Возвращает

Details Максимальная производительность достигается при размещении входных, выходных и временных массивов в разных банках памяти. Массивы dst1 и dst2 могут находиться в одном банке. Макс производительность на 64-р. слово результата = 2.1 такта (при size=10240 байт) и 2.5 такта (при size=4096 байт)

6.196 nmppSplitEco

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. |

Возвращает

Details Максимальная производительность достигается при размещении входных, выходных массивов в разных банках памяти. Массивы dst1 и dst2 могут находиться в одном банке. Макс производительность на 64-р. слово результата = 2.14 такта (при size=10240 байт) и 2.6 такта (при size=4096 байт)

6.197 Типы данных

Группы

- [Типы векторных данных](#)
- [Типы скалярных данных](#)

6.197.1 Подробное описание

6.198 Векторные функции

Группы

- [Функции поддержки](#)
- [Инициализация и копирование](#)
- [Арифметические операции](#)
- [Логические и бинарные операции](#)
- [Операции сравнения](#)
- [Переупорядочивание и сортировка](#)

6.198.1 Подробное описание

6.199 Матричные функции

Группы

- [Инициализация и копирование](#)
- [Функции поддержки](#)
- [Векторно-матричные операции](#)

6.199.1 Подробное описание

6.200 Функции обработки сигналов

Группы

- [Свертка](#)
- [Масочная фильтрация](#)
- [Изменение размеров](#)
- [Быстрое преобразование Фурье](#)

6.200.1 Подробное описание

6.201 Функции обработки изображений

Группы

- [Floodfill](#)

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

- [Переупорядочивание изображений](#)
- [Арифметические действия](#)
- [Масочная фильтрация](#)
- [Инициализация и копирование](#)
- [Функции поддержки](#)
- [Функции графического вывода текста](#)

6.201.1 Подробное описание

6.202 Скалярные функции

Группы

- [Инициализация](#)
- [Integer operations](#)
- [Fix-point 64](#)
- [Fix-point 32](#)
- [Арифметические операции](#)
- [Функции деинтерлейсинга](#)

6.202.1 Подробное описание

6.202.1.1 Введение

Назначением данной библиотеки является предоставление базовых операций по работе со скалярными данными для процессора NM6403, NM6404, NM6405.

В состав библиотеки входят арифметические, тригонометрические функции, функции для работы с данными в формате с фиксированной точкой.

Библиотека предназначена для быстрой разработки эффективных пользовательских программ на языке высокого уровня(C++).

Назначением данной библиотеки является предоставление базовых операций обработки изображений для процессора NM6403, NM6404, NM6405. В состав библиотеки входят функции двумерной фильтрации, арифметические действия и цветовые преобразования. Библиотека предназначена для быстрой разработки эффективных пользовательских программ на языке высокого уровня с использованием преимуществ архитектуры данного процессора.

Функции библиотеки имеют C++ интерфейс. Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессора NM6403.

Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Назначением данной библиотеки является предоставление базовых операций по обработке матриц для процессорах NM6403, NM6404, NM6405. В состав библиотеки входят арифметические операции над матрицами. Библиотека предназначена для быстрой разработки эффективных пользовательских программ как на языке высокого уровня(C++).

Функции библиотеки имеют C++ интерфейс. Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессора NM6403.

Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Назначением данной библиотеки является предоставление базовых функций по обработке сигналов для процессоров NM6403,NM6404,NM6405. В состав библиотеки входят функции одномерной КИХ

фильтрации, нелинейной фильтрации, передискретизации. Библиотека предназначена для быстрой разработки эффективных пользовательских программ как на языке высокого уровня(C++).

Функции библиотеки имеют C++ интерфейс. Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессоров NM6403.

Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Назначением данной библиотеки является предоставление базовых операций по обработке одномерных массивов (векторов) для процессоров NM6405, NM6406, систем на 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 x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Функции векторного ядра библиотеки

Функции различных библиотек: nmplv, nmpls, nmpli, nmplm и др. , имеющие C++ интерфейсы, в своей реализации используют вызовы функций ядра. Функции ядра не имеют C++ интерфейса. Их вызов возможен только из ассемблера процессора NeuroMatrix. Передача параметров и настройка функций производится через регистры.

Одна и та же функция ядра может использоваться при реализации одной

или нескольких функций библиотеки. Функции ядра также могут быть использованы для реализации пользовательских функций. Использование функций ядра позволяет минимизировать время разработки, уменьшить размер кода и получить максимальную производительность.

6.203 Базовые регистровые функции библиотеки

Группы

- [Элементарные функции](#)
- [функции взвешенного суммирования](#)
- [Целевые функции](#)

6.203.1 Подробное описание

6.204 контроль переполнения

Классы

- class `nmmtr< T >`
- class `nmmtrpack< T >`
- class `nmvecpack< T >`
- class `vec< T >`

Макросы

- `#define GetVec getvec`

Функции

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

6.204.1 Подробное описание

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

—реализации библиотеки на РС производит контроль переполнения с выдачей сообщения об ошибке пользователю библиотеки. Для этой цели определены шаблонные классы для вектора, матрицы и скалярного числа, позволяющие производить базовые операции над их элементами.

6.204.2 Макросы

6.204.2.1 GetVec

```
#define GetVec getvec
```

Класс матриц.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};
int Res [10];
```

```
mtr<int> AA0(3,3);
vec<int> A0(3);
scalar<int> a0(2);
```

```
mtr<int> BB0(3,3);
mtr<int> CC0(3,3);
vec<int> C0(3);
```

```
BB0.SetData a( Test);
```

```
BB0=AA0;
A0 =AA0[2];
a0 =AA0[2][2];
```

```
CC0=AA0+BB0;
CC0=AA0*a0;
C0 =AA0*A0;
CC0=AA0*BB0;
```

См. определение в файле `tmatrix.h` строка 60

6.204.3 Функции

6.204.3.1 operator<<()

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

Класс матриц.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};  
int Res [10];  
  
mtr<int> AA0(3,3);  
vec<int> A0(3);  
scalar<int> a0(2);  
  
mtr<int> BB0(3,3);  
mtr<int> CC0(3,3);  
vec<int> C0(3);  
  
BB0.SetData( Test);  
  
BB0=AA0;  
A0 =AA0[2];  
a0 =AA0[2][2];  
  
CC0=AA0+BB0;  
CC0=AA0*a0;  
C0 =AA0*A0;  
CC0=AA0*BB0;
```

См. определение в файле nmtlio.h строка 64

6.205 Элементарные функции

Группы

- [Vec_0_sub_data](#)
- [Vec_activate_data](#)
- [Vec_activate_data_add_0](#)
- [Vec_activate_data_xor_data](#)
- [Vec_activate_data_add_ram](#)
- [Vec_affo](#)
- [Vec_data](#)
- [Vec_data_add_ram](#)
- [Vec_data_and_ram](#)
- [Vec_data_or_ram](#)
- [Vec_data_sub_ram](#)
- [Vec_data_xor_ram](#)
- [Vec_And](#)
- [Vec_Mask](#)
- [Vec_Or](#)
- [Vec_Xor](#)
- [Vec_Add](#)
- [Vec_Sub](#)
- [Vec_not_data](#)
- [Vec_ram](#)
- [Vec_ram_sub_data](#)
- [Vec_vsum_activate_data_0](#)

6.205.1 Подробное описание

6.206 функции взвешенного суммирования

Группы

- [Vec_ClipMul2D2W8_AddVr](#)
- [Vec_ClipMulNDNW2_AddVr](#)
- [Vec_ClipMulNDNW4_AddVr](#)
- [Vec_ClipMulNDNW8_AddVr](#)
- [Vec_Mul2D2W1_AddVr](#)
- [Vec_Mul2D2W2_AddVr](#)
- [Vec_Mul2D2W4_AddVr](#)
- [Vec_Mul2D2W8_AddVr](#)
- [Vec_Mul3D3W2_AddVr](#)
- [Vec_Mul3D3W8_AddVr](#)
- [Vec_Mul4D4W2_AddVr](#)
- [Vec_MulVN_AddVN](#)
- [Vec_vsum_data_0](#)
- [Vec_vsum_data_vr](#)
- [Vec_vsum_shift_data_0](#)
- [Vec_vsum_shift_data_vr](#)
- [Vec_vsum_shift_data_afifo](#)

6.206.1 Подробное описание

6.207 Целевые функции

Группы

- [Vec_Add_VV_shift](#)
- [Vec_data_add_afifo](#)
- [Vec_FilterCoreRow2](#)
- [Vec_FilterCoreRow4](#)
- [Vec_FilterCoreRow8](#)
- [Vec_Abs](#)
- [Vec_ClipExt](#)
- [Vec_IncNeg](#)
- [Vec_SubAbs](#)
- [Vec_SubVN_Abs](#)
- [Vec_Swap](#)
- [Vec_MUL_2V4toW8_shift](#)
- [Vec_MUL_2V8toW16_shift](#)
- [Vec_vsum_data_afifo](#)
- [Vec_CompareMinV](#)

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec_CompareMaxV](#)

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec_DupValueInVector8](#)

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec_DupValueInVector16](#)

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec_BuildDiagWeights8](#)

Построение диагональной матрицы весовых коэффициентов (8x8).

- [Vec_BuildDiagWeights16](#)

Построение диагональной матрицы весовых коэффициентов (16x16).

- [Vec_MaxVal_v8nm8s](#)

Поиск максимума в 8 байтах

- [Vec_MaxVal_v4nm16s](#)

Поиск максимума в 4-х 16р. элементах

- [Vec_MaxVal](#)

Поиск максимумов в колонках матрицы SrcMtr1.

- [Vec_MinVal_v8nm8s](#)

Поиск минимума в 8 байтах

- [Vec_MinVal_v4nm16s](#)

Поиск минимума в 4-х 16р. элементах

- [Vec_MinVal](#)

Поиск минимумов в колонках матрицы SrcMtr1.

- [Vec_AccMul1D1W32_AddVr](#)

Умножение с накоплением

Действие функции эквивалентно следующим псевдоинструкциям:

6.207.1 Подробное описание

6.208 Vec_0_sub_data

Функции

- void vec_0_sub_data (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.208.1 Подробное описание

Ядро функции nmppsNeg().

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.209 Vec_activate_data

Функции

- void vec_activate_data (nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.209.1 Подробное описание

Ядро функции nmppsCmpLt0_().

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|------|--|
| f1cr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.210 Vec_activate_data_add_0

Функции

- void vec_activate_data_add_0 (nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.210.1 Подробное описание

Функция производит арифметическую активацию.

Ядро функции nmppsClipPowC_().

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|------|--|
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.211 Vec_activate_data_xor_data

Функции

- void vec_activate_data_xor_data (nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.211.1 Подробное описание

Функция позволяет вычислить приближенное значение модуля.

Ядро функции nmppsAbs1().

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|------|--|
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.212 Vec_activate_data_add_ram

Функции

- void vec_activate_data_add_ram (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.212.1 Подробное описание

Функция выполняет арифметическую активацию с прибавлением константы.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1]
rep N data=[ar0++gr0] with activate data+ram;
rep N [ar6++gr6] = afifo;
```

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на 64р. слово |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.213 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)

6.213.1 Подробное описание

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

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar4];
rep N data = [ar0++gr0] with data + 0;
rep N data = [ar1++gr1] with data + aifo;
rep N with mask ram,shift aifo,0;
rep N [ar6++gr6] = aifo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на строки |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на 64р. слово (маска) |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.214 Vec_afifo

Функции

- void vec_afifo (nmreg ar0, nmreg gr5, nmreg ar6, nmreg gr6)

6.214.1 Подробное описание

Функция служит для заполнения массива константой.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1 data=[ar0] with data;  
rep 1*N [ar6++gr6]=afifo with afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на 64р. слово |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.215 Vec_data

Функции

- void vec_data (nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.215.1 Подробное описание

Ядро функции nmppsCopy_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N [ar6++gr6]=aifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.216 Vec_data_add_afifo

Функции

- void vec_data_add_afifo (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

6.216.1 Подробное описание

Ядро функции nmppsSum().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1  data=vfalse;
.repeat N;
rep 1  data=[ar0++gr0] with data + afifo; (rep1 N times)
.endrepeat;
rep 1  [ar6]=afifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на 64р. слово |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0.

6.217 Vec_data_add_ram

Функции

- void vec_data_add_ram (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.217.1 Подробное описание

Ядро функции nmppsAddC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data+ram;
rep N [ar6++gr6]=aiffo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr stride |
| ar1 | указатель на 64р. слово-константу |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.218 Vec_data_and_ram

Функции

- void vec_data_and_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.218.1 Подробное описание

Ядро функции nmppsAndC_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram =[ar1];
rep N data=[ar0++gr0] with data and ram;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на 64р. слово-константу |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.219 Vec_data_or_ram

Функции

- void vec_data_or_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.219.1 Подробное описание

Ядро функции nmppsOrC_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data or ram;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на 64р. слово-константу |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.220 Vec_data_sub_ram

Функции

- void vec_data_sub_ram (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.220.1 Подробное описание

Ядро функции nmppsSubC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data-ram;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| ar1 | указатель на 64р. слово-константу |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.221 Vec_data_xor_ram

Функции

- void vec_data_xor_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.221.1 Подробное описание

Ядро функции nmppsXorC_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar1];
rep N data = [ar0++gr0] with data xor ram;
rep N [ar6++gr6] = afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на 64р. слово-константу |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.222 Vec_FilterCoreRow2

Функции

- void vec_FilterCoreRow2 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

6.222.1 Подробное описание

6.223 Vec_FilterCoreRow4

Функции

- void vec_FilterCoreRow4 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

6.223.1 Подробное описание

6.224 Vec_FilterCoreRow8

Функции

- void vec_FilterCoreRow8 ([nmreg](#) ar0, [nmreg](#) ar4, [nmreg](#) ar6, [nmreg](#) gr1, [nmreg](#) gr4, [nmreg](#) gr6)

6.224.1 Подробное описание

6.225 Vec_And

Функции

- void vec_And (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.225.1 Подробное описание

Ядро функции nmppsAnd_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data and afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.226 Vec_Mask

Функции

- void vec_Mask (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar2, nmreg gr2, nmreg gr5, nmreg ar6, nmreg gr6)

6.226.1 Подробное описание

Ядро функции nmppsMaskV_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with data;
rep N data=[ar2++gr2] with mask data,ram,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar2 | указатель на столбец SrcMtr2 (маска) |
| gr2 | SrcMtr3 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.227 Vec_Or

Функции

- void vec_Or (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.227.1 Подробное описание

Ядро функции nmppsOr_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data or aifo;
rep N [ar6++gr6]=aifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.228 Vec_Xor

Функции

- void vec_Xor (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.228.1 Подробное описание

Ядро функции nmppsXor_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data xor affo;
rep N [ar6++gr6]=affo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.229 Vec_Abs

Функции

- void vec_Abs (nmreg nb1, nmreg sb, nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.229.1 Подробное описание

Ядро функции nmppsAbs().

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.230 Vec_Add

Функции

- void vec_Add (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.230.1 Подробное описание

Ядро функции nmppsAdd().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data + aifo;
rep N [ar6++gr6]=aifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.231 Vec_ClipExt

Функции

- void vec_ClipExt (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg ar2, nmreg ar3, nmreg gr5, nmreg ar6, nmreg gr6)

6.231.1 Подробное описание

Ядро функции nmppsClipCC_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar1];
rep N data = [ar0++gr0] with data-ram;
rep N data = [ar2]          with activate afifo+data;
rep N data = [ar3]          with activate afifo-data;
rep N [ar6++gr6] = afifo;
```

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| ar1 | указатель на 64р. слово-константу |
| ar2 | указатель на 64р. слово-константу |
| ar3 | указатель на 64р. слово-константу |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

nmppsClipCC_

6.232 Vec_ClipMul2D2W8_AddVr

Функции

- void vec_ClipMul2D2W8_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.232.1 Подробное описание

Взвешенное умножение двух массивов с накоплением и активацией

Ядро функции nmppsClipPowC_RShift_Convert_AddC_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data =[ar1++gr1] with vsum ,activate data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| vr | константа для суммирования |
| ar0 | SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | 2 матрицы весовых коэффициентов по 8 64р. слов |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

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

6.233 Vec_ClipMulNDNW2_AddVr

Функции

- void vec_ClipMulNDNW2_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.233.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацией

Ядро функции SIG_Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
ar2=ar0;

gr2=[ar1++];
ar0=ar2+gr2;
rep 2 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;

.repeat K-1;
gr2=[ar1++];
ar0=ar2+gr2;
rep 2 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,afifo;
.endrepeat;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|------|---|
| nb1 | задает разбиение на колонки |
| sb | задает разбиение на 2 строки (sb=2) |
| flcr | задает функцию активации |
| vr | константа для суммирования |
| ar0 | задает базовый адрес для входных массивов (как правило адрес первого массива) |
| gr0 | шаг чтения входного массива stride for input arrays |
| ar1 | массив адресных смещений входных массивов относительно ar0 |
| gr1 | количество массив - K |
| ar4 | массив из K матриц весовых коэффициентов по 2 64р. слов |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,gr1,ar2,gr2,ar3,gr3,ar4,ar6,gr7.

6.234 Vec_ClipMulNDNW4_AddVr

Функции

- void vec_ClipMulNDNW4_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.234.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацией

Ядро функции SIG_Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
ar2=ar0;

gr2=[ar1++];
ar0=ar2+gr2;
rep 4 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;

.repeat K-1;
gr2=[ar1++];
ar0=ar2+gr2;
rep 4 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,afifo;
.endrepeat;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|------|---|
| nb1 | задает разбиение на колонки |
| sb | задает разбиение на 4 строки (sb=20002h) |
| flcr | задает функцию активации |
| vr | константа для суммирования |
| ar0 | задает базовый адрес для входных массивов (как правило адрес первого массива) |
| gr0 | stride for input arrays |
| ar1 | массив адресных смещений входных массивов относительно ar0 |
| gr1 | количество массив - K |
| ar4 | массив из K матриц весовых коэффициентов по 4 64р. слов |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,gr1,ar2,gr2,ar3,gr3,ar4,ar6,gr7.

6.235 Vec_ClipMulNDNW8_AddVr

Функции

- void vec_ClipMulNDNW8_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.235.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацией

Ядро функции SIG_Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
ar2=ar0;

gr2=[ar1++];
ar0=ar2+gr2;
rep 8 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;

.repeat K-1;
gr2=[ar1++];
ar0=ar2+gr2;
rep 8 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,affo;
.endrepeat;
rep N [ar6++gr6]=affo;
```

Аргументы

| | |
|------|---|
| nb1 | задает разбиение на колонки |
| sb | задает разбиение на 8 строки (sb=2020202h) |
| flcr | задает функцию активации |
| vr | константа для суммирования |
| ar0 | \ задает базовый адрес для входных массивов (как правило адрес первого массива) \ set base address for input arrays |
| gr0 | stride for input arrays |
| ar1 | массив адресных смещений входных массивов относительно ar0 |
| gr1 | \ количество массив - K \ number of arrays - K |
| ar4 | массив из K матриц весовых коэффициентов по 8 64р. слов |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на результирующий столбец |
| gr6 | межстрочный шаг для ar6 |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,gr1,ar2,gr2,ar3,gr3,ar4,ar6,gr7.

6.236 Vec_IncNeg

Функции

- void vec_IncNeg (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.236.1 Подробное описание

Увеличивает отрицательные числа на 1.

Применяется в nmppsDivC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar0++gr0] with activate data;
rep N          with ram - aifo;
rep N [ar6++gr6] = aifo;
```

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.237 Vec_Mul2D2W1_AddVr

Функции

- void vec_Mul2D2W1_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.237.1 Подробное описание

Ядро функции nmppsConvert_64s(nm64s* ,nm32s* ,int).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 1 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|---|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение в 1 строку |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на 2 матрицы весовых коэффициентов по 2 строки в каждой |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.238 Vec_Mul2D2W2_AddVr

Функции

- void vec_Mul2D2W2_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.238.1 Подробное описание

Ядро функции nmppsConvert_32s(nm32s* ,nm16s* ,int).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|---|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 2 строки |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на 2 матрицы весовых коэффициентов по 2 строки в каждой |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.239 Vec_Mul2D2W4_AddVr

Функции

- void vec_Mul2D2W4_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

6.239.1 Подробное описание

Ядро функции nmppsConvert_16s(nm16s* pSrcVec, nm8s* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 4 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 4 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|------|---|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 4 строки |
| flcr | задает функцию активации |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на 2 матрицы весовых коэффициентов по 4 строки в каждой |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.240 Vec_Mul2D2W8_AddVr

Функции

- void vec_Mul2D2W8_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.240.1 Подробное описание

Применяется в MTR_Cоруua().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 8 wffo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 8 wffo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,affo;
rep N [ar6++gr6]=affo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на 2 матрицы весовых коэффициентов по 8 строк в каждой |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.241 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)

6.241.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar2++gr2] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|---|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 2 строки |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar2 | указатель на столбец SrcMtr3 |
| gr2 | SrcMtr3 stride |
| ar4 | указатель на 3 матрицы весовых коэффициентов по 2 строки в каждой |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.242 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)

6.242.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data=[ar2++gr2] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar2 | указатель на столбец SrcMtr3 |
| gr2 | SrcMtr3 stride |
| ar4 | указатель на 3 матрицы весовых коэффициентов по 8 строк в каждой |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.243 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)

6.243.1 Подробное описание

Ядро функции nmppsConvert_32s(nm32s* pSrcVec, nm8s* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar2++gr2] with vsum ,data,afifo;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar3++gr3] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|---|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 2 строки |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar2 | указатель на столбец SrcMtr3 |
| gr2 | SrcMtr3 stride |
| ar3 | указатель на столбец SrcMtr4 |
| gr3 | SrcMtr4 stride |
| ar4 | указатель на 4 матрицы весовых коэффициентов по 2 строки в каждой |
| gr4 | дублирует nb1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.244 Vec_MulVN_AddVN

Функции

- void vec_MulVN_AddVN (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.244.1 Подробное описание

Ядро функции MTR_MulC_AddVsVc().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,data, vr;
rep N data=[ar1++gr1] with affo+data;
rep N [ar6++gr6]=affo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.245 Vec_Sub

Функции

- void vec_Sub (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.245.1 Подробное описание

Ядро функции nmppsSub().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data - affo;
rep N [ar6++gr6]=affo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.246 Vec_SubAbs

Функции

- void vec_SubAbs (nmreg nb1, nmreg sb, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

6.246.1 Подробное описание

Ядро функции nmppsAbsDiff().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data
rep N data=[ar1++gr1] with afifo-data;
rep N [ar4],ram =afifo with activate afifo;
rep N with vsum afifo,ram,ram;
rep N [ar6++gr6]= afifo;
```

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на временный буфер (1 64р. слово) |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.247 Vec_SubVN_Abs

Функции

- void vec_SubVN_Abs (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

6.247.1 Подробное описание

Ядро функции mtr_SubMV_Abs().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1]
rep N data=[ar0++gr0] with data-ram;
rep N [ar2++]=afifo with activate afifo;
// ar2,ar5- internal pointers to temporary buffer size of long[32]
rep N data=[ar5++] with vsum afifo,data,data;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| ar1 | указатель на маску (1 64p. слово) |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.248 Vec_Swap

Функции

- void vec_Swap (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

6.248.1 Подробное описание

Ядро функции mtr_SubVN_Abs().

Функция осуществляет два одновременных копирования:

[ar0++gr0] => [ar4++gr4]

[ar1++gr1] => [ar6++gr6]

если ar6=ar0,gr6=gr0, ar4=ar1,gr4=gr1

то выполняется перестановка двух векторов

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with data;
rep N [ar6++gr6]=afifo with ram;
rep N [ar4++gr4]=afifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar4 | указатель на столбец DstMtr1 |
| gr4 | DstVec1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr2 |
| gr6 | DstVec2 stride |

Restrictions:

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

6.249 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)

6.249.1 Подробное описание

Ядро функции SIG_ResizeDown2(nm8u7b* pSrcVec, nm8u7b* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar3];
rep 8 wifo=[ar5++],ftw,wtw;
rep N data =[ar0++gr0],ftw,wtw with vsum ,data,0;
rep N data =[ar1++gr1]          with vsum ,data,afifo;
rep N          with mask ram,shift afifo,0;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки |
| sb | задает разбиение на 4 строки |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar3 | указатель на 64р. маску |
| gr4 | дублирует nb1 |
| ar5 | указатель на матрицу весовых коэффициентов (16 64р.слов) |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.250 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)

6.250.1 Подробное описание

Ядро функции SIG_ResizeDown2(nm16u15b* pSrcVec, nm16u15b* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar3];
rep 16 wfifo=[ar5++],ftw,wtw;
rep N data=[ar0++gr0],ftw,wtw with vsum ,data,0;
rep N data=[ar1++gr1]          with vsum ,data,afifo;
rep N                               with mask ram,shift afifo,0;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки |
| sb | задает разбиение на 2 строки |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| ar3 | указатель на 64р. маску |
| gr4 | дублирует nb1 |
| ar5 | указатель на матрицу весовых коэффициентов (16 64р.слов) |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.251 Vec_not_data

Функции

- void vec_not_data (nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.251.1 Подробное описание

Ядро функции nmppsNot_ ().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with not data;
rep N [ar6++gr6]=aifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.252 Vec_ram

Функции

- void vec_ram (nmreg ar0, nmreg gr5, nmreg ar6, nmreg gr6)

6.252.1 Подробное описание

Ядро функции nmppsSet_().

Функция служит для заполнения массива константой.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0] with data;  
rep N [ar6++gr6]=aifo;
```

Аргументы

| | |
|-----|--|
| ar0 | указатель на столбец SrcMtr1 |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.253 Vec_ram_sub_data

Функции

- void vec_ram_sub_data (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

6.253.1 Подробное описание

Ядро функции nmppsSubCRev().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with ram-data;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-----|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.254 Vec_vsum_activate_data_0

Функции

- void vec_vsum_activate_data_0 (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

6.254.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1 | указатель на столбец SrcMtr2 |
| gr1 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.255 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)

6.255.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.256 Vec_vsum_data_afifo

Функции

- void vec_vsum_data_afifo (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

6.256.1 Подробное описание

Используется в nmppsSum(nm1* pSrcVec, void* pTmpBuf, int nSize)

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1  data=vfalse;
.repeat N;
rep 1  data=[ar0++gr0] with vsum ,data,afifo;
.endrepeat;
rep 1  [ar6]=afifo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на строки |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0.

6.257 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)

6.257.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,data,vr;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.258 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)

6.258.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

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

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц $N = [0,1,2...31,32,33,...]$ |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.259 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)

6.259.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,shift data, vr;
rep N [ar6++gr6]=afifo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.260 Vec_vsum_shift_data_afifo

Функции

- void vec_vsum_shift_data_afifo (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

6.260.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1  data=vfalse;
rep 1*N data=[ar0++gr0] with vsum , shift data,afifo; (rep1 N times)
rep 1  [ar6]=afifo;
```

Аргументы

| | |
|-------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| sb | задает разбиение на 8 строк |
| flcr | задает функцию активации |
| woper | в рабочей матрице должны быть загружены весовые коэффициенты |
| ar0 | указатель на столбец SrcMtr |
| gr0 | SrcMtr stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0.

6.261 Vec_CompareMinV

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec_CompareMinV (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr3, nmreg gr5, nmreg ar6, nmreg gr6)

6.261.1 Подробное описание

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with ram-data;
rep N      with activate aifo;
rep N data=[ar3++gr3] with mask aifo, ram, data;
rep N [ar6++gr6]=aifo;
```

\param nb1

задает разбиение на колонки (необходимо wtw)

Аргументы

| | |
|---------|--|
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1,ar3 | указатель на столбец SrcMtr2 |
| gr1,gr3 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.262 Vec_CompareMaxV

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec_CompareMaxV (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr3, nmreg gr5, nmreg ar6, nmreg gr6)

6.262.1 Подробное описание

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with ram-data;
rep N      with activate aifo;
rep N data=[ar3++gr3] with mask aifo, data, ram;
rep N [ar6++gr6]=aifo;
```

\param nb1

задаёт разбиение на колонки (необходимо wtw)

Аргументы

| | |
|---------|--|
| flcr | задаёт функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr1 stride |
| ar1,ar3 | указатель на столбец SrcMtr2 |
| gr1,gr3 | SrcMtr2 stride |
| gr5 | Высота матриц N = [0,1,2...31,32,33,...] |
| ar6 | указатель на столбец DstMtr |
| gr6 | DstMtr stride |

Restrictions:

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

6.263 Vec_DupValueInVector8

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec_DupValueInVector8 (nmreg ar1, nmreg gr1)

6.263.1 Подробное описание

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

```
gr1 = gr1 and 0FFh;  
[ar1] = gr1 + (gr1 << 8) + (gr1 << 16) + (gr1 << 24) +  
        (gr1 << 32) + (gr1 << 40) + (gr1 << 48) + (gr1 << 56).  
ar1 += 2;
```

\param ar1

Адрес 64-х битового вектора.

Аргументы

| | |
|-----|-------------------|
| gr1 | Значение (8 бит). |
|-----|-------------------|

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

6.264 Vec_DupValueInVector16

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec_DupValueInVector16 (nmreg ar1, nmreg gr1)

6.264.1 Подробное описание

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

```
gr1 = gr1 and 0FFFFh;
[ar1] = gr1 + (gr1 << 16) + (gr1 << 32) + (gr1 << 48).
ar1 += 2;
```

\param ar1

Адрес 64-х битового вектора.

Аргументы

| | |
|-----|-------------------|
| gr1 | Значение (8 бит). |
|-----|-------------------|

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

6.265 Vec_BuildDiagWeights8

Построение диагональной матрицы весовых коэффициентов (8x8).

Функции

- void vec_BuildDiagWeights8 (nmreg ar1, nmreg gr1)

6.265.1 Подробное описание

Построение диагональной матрицы весовых коэффициентов (8x8).

Аргументы

| | |
|-----|---|
| ar1 | Адрес 64-х буфера весовых коэффициентов (8x64 бит). |
| gr1 | Значение (8 бит). |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

6.266 Vec_BuildDiagWeights16

Построение диагональной матрицы весовых коэффициентов (16x16).

Функции

- void vec_BuildDiagWeights16 (nmreg ar1, nmreg gr1)

6.266.1 Подробное описание

Построение диагональной матрицы весовых коэффициентов (16x16).

Аргументы

| | |
|-----|---|
| ar1 | Адрес 64-х буфера весовых коэффициентов (4x64 бит). |
| gr1 | Значение (16 бит). |

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

6.267 Vec_MaxVal_v8nm8s

Поиск максимума в 8 байтах

Функции

- void vec_MaxVal_v8nm8s (nmreg ar0, nmreg gr7)

6.267.1 Подробное описание

Поиск максимума в 8 байтах

Аргументы

| | |
|-----|------------------|
| ar0 | Адрес 64р. слова |
|-----|------------------|

Возвращаемые значения

| | |
|-----|--------------------|
| gr7 | Максимум из 8 байт |
|-----|--------------------|

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.

6.268 Vec_MaxVal_v4nm16s

Поиск максимума в 4-х 16р. элементах

Функции

- void vec_MaxVal_v4nm16s ([nmreg](#) ar0, [nmreg](#) gr7)

6.268.1 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.

6.269 Vec_MaxVal

Поиск максимумов в колонках матрицы SrcMtr1.

Функции

- void vec_MaxVal (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr5, nmreg ar6)

6.269.1 Подробное описание

Поиск максимумов в колонках матрицы SrcMtr1.

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| flcr | задает функцию активации |

\param ar0

указатель на столбец SrcMtr1

Аргументы

| | |
|-----|--|
| gr0 | SrcMtr stride |
| ar4 | указатель на временный массив размером nm64s[64] |
| gr5 | Высота матрицы SrcMtr1 N = [32,64,...] |

Возвращаемые значения

| | |
|-----|--|
| ar6 | указатель на 64р. слово результатов (максимумов) |
|-----|--|

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,gr0,ar4,ar3,ar6,gr7.

6.270 Vec_MinVal_v8nm8s

Поиск минимума в 8 байтах

Функции

- void vec_MinVal_v8nm8s ([nmreg](#) ar0, [nmreg](#) gr7)

6.270.1 Подробное описание

Поиск минимума в 8 байтах

Аргументы

| | |
|-----|------------------|
| ar0 | Адрес 64р. слова |
|-----|------------------|

Возвращаемые значения

| | |
|-----|--------------------|
| gr7 | Максимум из 8 байт |
|-----|--------------------|

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.

6.271 Vec_MinVal_v4nm16s

Поиск минимума в 4-х 16р. элементах

Функции

- void vec_MinVal_v4nm16s ([nmreg](#) ar0, [nmreg](#) gr7)

6.271.1 Подробное описание

Поиск минимума в 4-х 16р. элементах

Аргументы

| | |
|-----|------------------|
| ar0 | Адрес 64р. слова |
|-----|------------------|

Возвращаемые значения

| | |
|-----|--------------------|
| gr7 | Максимум из 8 байт |
|-----|--------------------|

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.

6.272 Vec_MinVal

Поиск минимумов в колонках матрицы SrcMtr1.

Функции

- void vec_MinVal (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr5, nmreg ar6)

6.272.1 Подробное описание

Поиск минимумов в колонках матрицы SrcMtr1.

Аргументы

| | |
|------|--|
| nb1 | задает разбиение на колонки (необходимо wtw) |
| flcr | задает функцию активации |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr stride |
| ar4 | указатель на временный массив размером nm64s[64] |
| gr5 | Высота матрицы SrcMtr1 N = [32,64,...] |

Возвращаемые значения

| | |
|-----|--|
| ar6 | указатель на 64р. слово результатов (максимумов) |
|-----|--|

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,gr0,ar4,ar3,ar6,gr7.

6.273 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)

6.273.1 Подробное описание

Умножение с накоплением

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 32 wfifo=[ar4++],ftw,wtw;
rep 32 wfifo data = [ar0++gr0] with vsum ,data,vr;
with gr5--;
with gr5--;
<Loop>
    rep 32 wfifo=[ar4++],ftw,wtw;
    rep 32 data=[ar0++gr0] with vsum ,data,afifo;
if <>0 goto Loop with gr5--;
rep 32 [ar6++gr6]=afifo;
```

\param nb1

задает разбиение на колонки (необходимо wtw)

Аргументы

| | |
|-----|---|
| sb | задает разбиение на 32 строки |
| vr | константа для суммирования |
| ar0 | указатель на столбец SrcMtr1 |
| gr0 | SrcMtr stride |
| ar4 | матрицы весовых коэффициентов |
| gr4 | дублирует nb1 |
| gr5 | кол-во итераций умножений с накоплением |

Возвращаемые значения

| | |
|-----|---|
| ar6 | указатель на столбец DstMtr, состоящий из 32 длинных слов |
|-----|---|

Аргументы

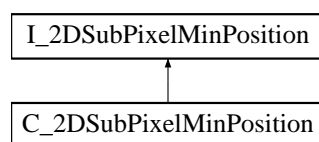
| | |
|-----|---------------|
| gr6 | DstMtr stride |
|-----|---------------|

Глава 7

Классы

7.1 Класс C_2DSubPixelMinPosition

Граф наследования: C_2DSubPixelMinPosition:



Открытые члены

- virtual void Find (float *S9, float &dx, float &dy)
- virtual void Release ()

7.1.1 Подробное описание

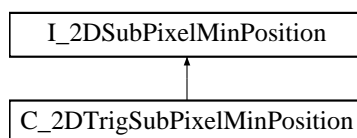
См. определение в файле isubpixel2dimpl.h строка 7

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h

7.2 Класс C_2DTrigSubPixelMinPosition

Граф наследования: C_2DTrigSubPixelMinPosition:



Открытые члены

- virtual void Find (float *S9, float &dx, float &dy)
- virtual void Release ()

Защищенные члены

- float S9Interpolation (float x, float y, float *S9)

Защищенные данные

- float Teta [8]

7.2.1 Подробное описание

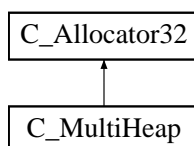
См. определение в файле isubpixel2dimpl.h строка 20

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h

7.3 Класс C_Allocator32

Граф наследования: C_Allocator32:



Открытые члены

- void * Allocate ()
- int Release ()

7.3.1 Подробное описание

См. определение в файле multiheap.h строка 330

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

7.4 Шаблон класса C_BoxImg< T >

Открытые члены

- C_BoxImg (C_MultiHeap &MultiHeap, int Width, int Height, int BorderHeight=0, int FillMode=BOX_IMG_FILL_FF)
- void Lock ()
- void Unlock ()
- void Fill (int FillMode)
- C_BoxImg (T *Data, int Width, int Height, int BorderHeight=0)
- void Init (T *Data, int Width, int Height, int BorderHeight=0)
- T * Addr (int y, int x)
- T * Allocate (C_MultiHeap &MultiHeap, int Width, int Height, int BorderHeight=0)
- T * Allocate (C_MultiHeap &MultiHeap)
- int Release ()
- __INLINE__ T * operator[] (int idx)

Открытые атрибуты

- int nWidth
- int nHeight
- int sizeBox
- int sizeData
- int nBorder
- T * pBox
- T * pData
- C_MultiHeap * pHeap

7.4.1 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

7.5 Шаблон класса C_BoxVec< T >

Открытые члены

- C_BoxVec (T *Data, int SizeData, int SizeBorder=0)
- C_BoxVec (C_MultiHeap &MultiHeap, int SizeData, int Border=0)
- T * Addr (int idx)
- int Assign (T *Data, int SizeData, int SizeBorder=0)
- T * Allocate (C_MultiHeap &MultiHeap, int SizeData, int Border=0)
- T * Allocate (C_MultiHeap &MultiHeap)
- int Release ()

Открытые атрибуты

- int sizeData
- int sizeBox
- int nBorder
- T * pBox
- T * pData
- [C_MultiHeap](#) * pHeap

7.5.1 Подробное описание

```
template<class T>
class C_BoxVec< T >
```

См. определение в файле multiheap.h строка 634

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

7.6 Класс C_Heap

класс - куча

```
#include <multiheap.h>
```

Открытые члены

- [C_Heap](#) (void *addrHeap, size_t size32Heap)
конструктор - создает кучу в указанной памяти
- void [Create](#) (void *addrHeap, size_t size32Heap)
создает кучу в указанной памяти
- int [IsMine](#) (void *addr)
устанавливает принадлежность к куче
- size_t [AllocateMaxAvail](#) ()
Возвращает объем свободной памяти в пуле в 32-х словах
- int * [Allocate](#) (size_t size32Buffer)
Выделяет буфер в куче
- int [ReleaseBuffer](#) ([S_BufferInfo](#) *pDelBuffer)
удаляет структуру буфера из списка
- int [Release](#) (void *p)
освобождает память по адресу
- void [Lock](#) (void *p)
блокирует указатель от удаления через Release.
- void [LockAll](#) ()
блокирует все указатели от удаления через Release.
- void [UnlockAll](#) ()
разблокирует все указатели для удаления через Release.
- void [Unlock](#) (void *p)
разблокирует все указатели для удаления через Release.
- void [ReleaseAll](#) ()
удаляет все указатели из кучи
- void [LockHeap](#) ()
Запрещает операции с кучей
- void [UnlockHeap](#) ()
Разрешает операции с кучей
- int [Check](#) ()

Открытые атрибуты

- [S_BufferInfo * pZeroBuffer](#)
< указатель на нулевой буфер в списке (с нулевым размером)
- [int * pHeapEnd](#)
< указатель на слово следующее за концом кучи
- [int size32HeapAvail](#)
< размер общей свободной памяти в куче
- [bool isHeapLocked](#)
< запрещает операции с кучей
- [int status](#)

7.6.1 Подробное описание

класс - куча

См. определение в файле multiheap.h строка 64

7.6.2 Методы

7.6.2.1 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

7.7 Шаблон класса C_Img< T >

Открытые члены

- [C_Img \(int nWidth, int nHeight, int nStride, int nBorder, void *\(*allocator32\)\(int\)\)](#)
- [C_Img \(T *pData, int nWidth, int nHeight, int nStride, int nBorder\)](#)
- [void Fill \(T color\)](#)

Открытые атрибуты

- int m_nBorder
- T * m_pContainer
- T * m_pData
- int m_nWidth
- int m_nHeight
- int m_nStride
- int m_nSize

7.7.1 Подробное описание

```
template<class T>
class C_Img< T >
```

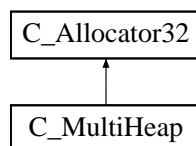
См. определение в файле isupport.h строка 37

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isupport.h

7.8 Класс C_MultiHeap

Граф наследования: C_MultiHeap:



Открытые члены

- C_MultiHeap (int Mode=ALLOCATE_FORWARD)
- int Error ()
- void Mode (int mode, void **legend=0)
- unsigned Rand ()
 - Генератор случайных чисел
- unsigned Rand (unsigned min, unsigned max)
- C_Heap & operator[] (int idxHeap)
- int CreateHeap (void *addrHeap, size_t size32Heap)
 - создает кучу по адресу указанного размера (полный размер со служебными данными)
- void * Allocate (size_t size32Buffer)
 - обходит кучи в заданном в AllocateMode порядке и выделяет память заданного размера
- void * Allocate (size_t size32Buffer, int nPriorHeap0, int nPriorHeap1=-1, int nPriorHeap2=-1, int nPriorHeap3=-1, int nPriorHeap4=-1, int nPriorHeap5=-1)
 - обходит кучи в заданном порядке и выделяет память заданного размера
- void * AllocateWith (size_t size32Buffer, void *addrInTheSameHeap)
 - выделяет массив в той же куче где и указатель

- int Which (void *addr)
Возвращает номер кучи к которой принадлежит адрес
- void Lock (void *addr)
- int Unlock (void *addr)
- int LockAll ()
- int UnlockAll ()
- int Release (void *addr)
- void ReleaseAll ()
Удвояет все назаблокированные указатели из куч
- void LockHeap (int idxHeap)
Запрещает операции Allocate и Release с кучей
- void UnlockHeap (int idxHeap)
Разрешает операции Allocate и Release с кучей
- int Check ()

Открытые атрибуты

- C_Heap pHeap [MAX_NUM_BANKS]
массив куч
- unsigned numHeaps
число проинициализированных куч
- unsigned numAllocateFails
число ошибок выделения куч
- unsigned AllocateMode
порядок обхода куч при поиске свободного места
- void ** pAllocateLegend
история номеров куч использованных в последних 8 Allocate.
- unsigned idxAllocateLegend
- long long allocateHistory

7.8.1 Подробное описание

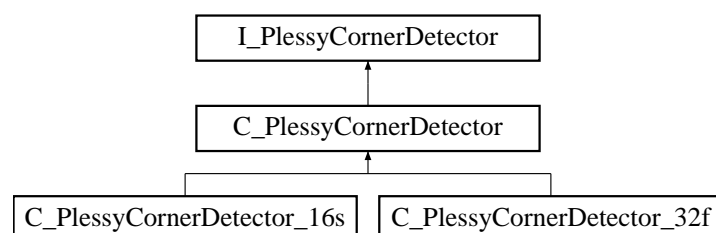
См. определение в файле multiheap.h строка 336

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

7.9 Класс C_PlessyCornerDetector

Граф наследования: C_PlessyCornerDetector:



Открытые члены

- virtual void Allocate (int w, int h, int ww)
- virtual void DeAllocate ()
- virtual void FindCorners (unsigned char *Picture, int w, int h, int ww, float *px, float *py, int &nc)
- virtual void Release ()
- virtual void SetThreshold (float _threshold)=0

Защищенные члены

- virtual void CountDer (unsigned char *Picture, int w, int h, int ww)
- virtual void CountPlessy (int w, int h, int ww)

Защищенные данные

- [I_2DSubPixelMinPosition](#) * SubPixelMinPosition
- short * fxi
- short * fyi
- short * Picture16
- float * SumSxxSyy
- float * Subxy
- float S9 [9]
- float threshold
- unsigned char * cres

7.9.1 Подробное описание

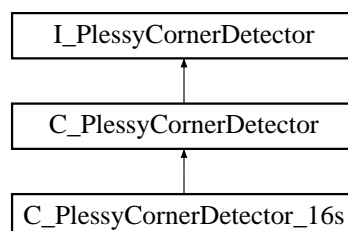
См. определение в файле iPlessyDetector.h строка 8

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iPlessyDetector.h

7.10 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

7.10.1 Подробное описание

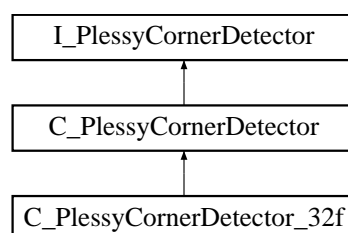
См. определение в файле iPlessyDetector.h строка 56

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iPlessyDetector.h

7.11 Класс C_PlessyCornerDetector_32f

Граф наследования:C_PlessyCornerDetector_32f:



Открытые члены

- virtual void Allocate (int w, int h, int ww)
- virtual void DeAllocate ()
- virtual void SetThreshold (float _threshold)

Защищенные члены

- virtual void CountPlessy (int w, int h, int ww)
- virtual void CountDer (unsigned char *Picture, int w, int h, int ww)

Защищенные данные

- float * fx
- float * fy
- float * sxx
- float * sxy
- float * syx
- float * Sxx
- float * Sxy
- float * Syy
- float * MulSxxSyy
- float * MulSxySxy
- float * cSubxy
- float threshold

7.11.1 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

7.12 Шаблон класса C_RingBuffer< T >

Открытые члены

- void Sleep (clock_t msec)
- C_RingBuffer (T *buffer, size_t count, t_bytecpy pushmemcpy, t_bytecpy popmemcpy)
- bool Init (T *buffer, size_t count, t_bytecpy pushmemcpy, t_bytecpy popmemcpy)
- __INLINE__ bool IsFull ()
- __INLINE__ bool IsEmpty ()
- __INLINE__ size_t PushAvail ()
- __INLINE__ size_t PopAvail ()
- T * Head ()
- T * Tail ()
- bool MoveHead (int numElements)
- bool MoveTail (int numElements)
- size_t Push (T *pSrcElements, size_t numElements, int ExitMode=EXIT_ON_COMPLETED)
- size_t Push (T pSrcElement)
- void [PushRequest](#) (T *pSrcElements, size_t numElements)
- bool isPushCompleted ()
- size_t Pop (T *pDstElements, size_t numElements, int ExitMode=EXIT_ON_COMPLETED)

Открытые атрибуты

- T * [data](#)
физический адрес кольцевого буфера входных данных
- 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 * [head_addr](#)
- size_t * [tail_addr](#)
- size_t [id](#)
- t_bytecpy [push_memcpy](#)
- t_bytecpy [pop_memcpy](#)
- memcpy_ptr [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(memcpy_ptr)/sizeof(int)]
резервные поля

7.12.1 Подробное описание

```
template<class T>
class C_RingBuffer< T >
```

См. определение в файле ringbuffer.h строка 33

7.12.2 Методы

7.12.2.1 PushRequest()

```
template<class T >
void C_RingBuffer< T >::PushRequest (
    T * pSrcElements,
    size_t numElements ) [inline]
```

!!! Head or head ??

См. определение в файле ringbuffer.h строка 203

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/ringbuffer.h

7.13 Шаблон класса C_RingBufferRemote< T >

Открытые члены

- C_RingBufferRemote (size_t ringbuffer_addr, t_bytecpy push_memcpy, t_bytecpy pop_memcpy)
- size_t GetHead ()
- size_t GetTail ()
- void SetHead ()
- void SetTail ()
- bool Init (size_t ringbuffer_addr, t_bytecpy push_memcpy, t_bytecpy pop_memcpy)
- __INLINE__ bool IsFull ()
- __INLINE__ bool IsEmpty ()
- __INLINE__ size_t GetWriteAvail ()
- __INLINE__ size_t GetReadAvail ()
- bool Push (int numElements)
- bool Pop (int numElements)
- size_t Push (T *pSrcElements, size_t numElements, int ExitMode=EXIT_ON_COMPLETED)
- size_t Pop (T *pDstElements, size_t numElements, int ExitMode=EXIT_ON_COMPLETED)
- size_t View (T *pDstElements, size_t numElements, int ExitMode=EXIT_ON_COMPLETED)

Открытые атрибуты

- size_t [data_addr](#)
физический адрес кольцевого буфера входных данных
- size_t [head_addr](#)
сколько элементов ОТ НАЧАЛА ПОТОКА код MASTER уже записал в буфер входных данных [заполняется MASTER].
- size_t [tail_addr](#)
сколько элементов ОТ НАЧАЛА ПОТОКА код SLAVE уже прочитал (обработал) [заполняется SLAVE].
- size_t [size](#)
размер кольцевого буфера входных данных (в элементах; гарантируется что это степень двойки)
- size_t [head](#)
сколько элементов ОТ НАЧАЛА ПОТОКА код MASTER уже записал в буфер входных данных [заполняется MASTER].
- size_t [tail](#)
сколько элементов ОТ НАЧАЛА ПОТОКА код SLAVE уже прочитал (обработал) [заполняется SLAVE].
- size_t [id](#)
сколько элементов ОТ НАЧАЛА ПОТОКА код SLAVE уже прочитал (обработал) [заполняется SLAVE].
- bool isConnected
- t_bytecpy push_memcpy
- t_bytecpy pop_memcpy
- t_memcpy dma_init
- size_t(* dma_check)()
- T * dma_ptr
- size_t dma_left
- size_t dma_size
- unsigned timeout
- unsigned time2sleep
- int [pad](#) [16-3-5 *sizeof(t_bytecpy)/sizeof(int)]
резервные поля

7.13.1 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

7.14 Шаблон класса C_WarpImg< T >

Открытые члены

- C_WarpImg (unsigned width, unsigned height, unsigned border, void *(*malloc32)(unsigned), void(*free32)(void *))
- C_WarpImg (unsigned width, unsigned height, unsigned border, void *buffer, int mode=IMG_↔ AT_BUFFER)
- T * addr (int x, int y)
- T * end ()

Открытые атрибуты

- unsigned nWidth
- unsigned nHeight
- unsigned warpHeight
- T * pWarp
- unsigned nWarpSize
- T * pImg
- unsigned nImgSize

7.14.1 Подробное описание

```
template<class T>
class C_WarpImg< T >
```

См. определение в файле warping.h строка 5

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/warpimg.h

7.15 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 коэффициенты фильтра и инициализирует внутреннюю структуру хранения коэффициентов в `pKernel`.
- void `Filter` (nmbits_in `*pSrcImg`, nmbits_out `*pDstImg`, int `nImgWidth`, int `nImgHeight`)
Функция одномерной фильтрации
- `~CIMG_FIR` ()
Освобождает динамическую область памяти `pKernel`.

Открытые атрибуты

- void `*(pfFree32)(void *)`
Указатель на функции освобождения памяти (`pKernel`)
- int `nKerWidth`
Ширина окна коэффициентов КИХ фильтра
- int `nKerHeight`
Высота окна коэффициентов КИХ фильтра
- `nm64s * pKernel`
Указатель на внутреннюю структуру коэффициентов
- int `nKernelSize`
Размер памяти необходимый для хранения внутренней структуры коэффициентов

7.15.1 Подробное описание

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

Класс КИХ фильтра

Template Parameters

| | |
|-------------------------|---|
| <code>nmbits_in</code> | Тип указывающий разрядность входного изображения. Допустимые типы : <code>nm8s, nm16s, nm32s, nm64s</code> |
| <code>nmbits_out</code> | Тип указывающий разрядность выходного изображения. Допустимые типы : <code>nm8s, nm16s, nm32s, nm64s</code> . Разрядность входного вектора не должна превышать разрядности выходного. |

Пример

См. определение в файле `iFilter.h` строка 302

7.15.2 Конструктор(ы)

7.15.2.1 `CIMG_FIR()`

```
template<class nbits_in , class nbits_out >
CIMG_FIR< nbits_in, nbits_out >::CIMG_FIR (
    int nKerWidth,
    int nKerHeight,
    void (*)(unsigned) malloc32_func,
    void (*)(void *) free32_func )
```

Конструктор КИХ фильтра

Выделяет область памяти под внутреннюю структуру коэффициентов

Аргументы

| | |
|---------------|---|
| nKerWidth | Ширина окна фильтра. nKerWidth=[3,5,7,...] |
| nKerHeight | Высота окна фильтра. nKerHeight=[1,3,5,7,...] |
| malloc32_func | указатель на функцию выделения динамической памяти 32-разрядными словами. |
| free32_func | указатель на функцию динамического освобождения памяти |

7.15.3 Методы

7.15.3.1 `Filter()`

```
template<class nbits_in , class nbits_out >
void CIMG_FIR< nbits_in, nbits_out >::Filter (
    nbits_in * pSrcImg,
    nbits_out * pDstImg,
    int nImgWidth,
    int nImgHeight )
```

Функция одномерной фильтрации

$$pDstImg[y][x] = \sum_{i=0}^{nKerHeight-1} \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerHeight/2][x+j-nKerWidth/2] \cdot pWeights[i][j], x =$$

Аргументы

| | |
|------------|---|
| pSrcImg | входное изображение |
| pDstImg | выходное изображение |
| nImgWidth | Ширина изображения к которому данный фильтр будет применен. Кратность согласно входному типу. |
| nImgHeight | Высота изображения (измеряется в пикселях). |

Предупреждения

При вычислении первых и последних $nKerHeight/2$ строк производится выход за границы входного массива `pSrcImg`. Для корректного поведения функции необходимо дополнительно резервировать поля размером не менее $nImgWidth(nKernHeight/2+1)$ нулевых элементов перед началом и в конце массива `pSrcImg`.

7.15.3.2 SetWeights()

```
template<class nmbits_in , class nmbits_out >
void* CIMG_FIR< nmbits_in, nmbits_out >::SetWeights (
    int * pWeights,
    int nImgWidth )
```

Загружает коэффициенты фильтра и инициализирует внутреннюю структуру хранения коэффициентов в `pKernel`.

Аргументы

| | |
|------------------------|---|
| <code>pWeights</code> | коэффициенты фильтра |
| <code>nImgWidth</code> | Ширина изображения к которому данный фильтр будет применен. Кратность согласно входному типу. |

Возвращает

указатель на внутреннюю структуру коэффициентов. 0- Если память под структуру не была выделена.

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmpli/iFilter.h`

7.16 Структура `ds_struct`

Открытые атрибуты

- `int nnSpot`
общее число отбракованных пятен (по данному признаку)
- `int nnPxl`
суммарное число пикселей в отбракованных пятнах
- `int dttSpot`
общее время обработки отбракованных пятен (в тактах процессора)

7.16.1 Подробное описание

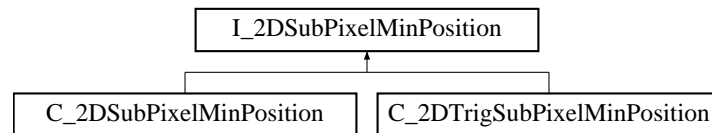
См. определение в файле `iFloodFill.h` строка 94

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmpli/iFloodFill.h`

7.17 Класс I_2DSubPixelMinPosition

Граф наследования:I_2DSubPixelMinPosition:



Открытые члены

- virtual void Find (float *S9, float &dx, float &dy)=0
- virtual void Release ()=0

7.17.1 Подробное описание

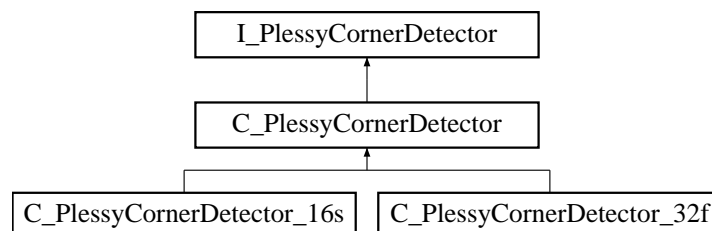
См. определение в файле isubpixel2d.h строка 7

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isubpixel2d.h

7.18 Класс I_PlessyCornerDetector

Граф наследования:I_PlessyCornerDetector:



Открытые члены

- virtual void Allocate (int w, int h, int ww)=0
- virtual void DeAllocate ()=0
- virtual void FindCorners (unsigned char *Picture, int w, int h, int ww, float *px, float *py, int &nc)=0
- virtual void Release ()=0
- virtual void SetThreshold (float _threshold)=0

7.18.1 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](#)

7.19 Структура int15in16x4

Открытые атрибуты

- unsigned long items

7.19.1 Подробное описание

См. определение в файле nmtype.h строка 234

Объявления и описания членов структуры находятся в файле:

- [D:/GIT/nmpp/include/nmtype.h](#)

7.20 Структура int30in32x2

Открытые атрибуты

- unsigned long items

7.20.1 Подробное описание

См. определение в файле nmtype.h строка 320

Объявления и описания членов структуры находятся в файле:

- [D:/GIT/nmpp/include/nmtype.h](#)

7.21 Структура int31in32x2

Открытые атрибуты

- unsigned long items

7.21.1 Подробное описание

См. определение в файле `nmtype.h` строка 300

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.22 Шаблон класса `mtr< T >`

Открытые члены

- `void origin (int y, int x)`
- `void setval (int y, int x, const T &val)`
- `void setvalx (int y, int x, const T &val)`
- `T getval (int y, int x) const`
- `T getvalx (int y, int x) const`
- `mtr (int nHeight, int nWidth, int nBorder=0)`
- `void resize (int nHeight, int nWidth, int nBorder=0)`
- `mtr (const mtr< T > &matr)`
- `void assign (T *Data, int nHeight, int nWidth, int nStride=0)`
- `mtr (T *Data, int nHeight, int nWidth, int nStride=0)`
- `mtr< T > & operator= (mtr< T > &matr)`
- `mtr< T > & operator= (T &val)`
- `T * operator[] (int row) const`
- `T & index (int idx)`
- `mtr< T > & operator*= (const T val)`
- `mtr< T > operator* (const T &val) const`
- `vec< T > operator* (const vec< T > &vect)`
- `mtr< T > operator* (const mtr< T > &matr)`
- `mtr< T > & operator+= (const T &val)`
- `mtr< T > & operator+= (const mtr< T > &matr)`
- `mtr< T > operator+ (const mtr< T > &matr) const`
- `mtr< T > & operator-= (const mtr< T > &matr)`
- `mtr< T > & operator-= (const T &val)`
- `mtr< T > operator- (const mtr< T > &matr) const`
- `mtr< T > operator- () const`
- `mtr< T > & operator/= (const T val)`
- `mtr< T > operator/ (const T val) const`
- `mtr< T > & operator>>= (const int shr)`
- `mtr< T > operator>> (const int shr) const`
- `mtr< T > & operator<<= (const int shl)`
- `mtr< T > operator<< (const int shl) const`
- `mtr< T > & operator &= (const T &val)`
- `mtr< T > & operator &= (const mtr< T > &matr)`
- `mtr< T > operator & (const mtr< T > &matr) const`
- `mtr< T > & operator|= (const mtr< T > &matr)`
- `mtr< T > operator| (const mtr< T > &matr) const`
- `mtr< T > & operator^= (const mtr< T > &matr)`
- `mtr< T > & operator^= (const T &val)`
- `mtr< T > operator^ (const mtr< T > &matr) const`
- `mtr< T > operator^ (const T &val) const`

- `mtr< T > operator~ ()`
- `void set (const T val)`
- `mtr< T > transpose ()`
- `mtr< T > & diag (T val)`
- `T * addr (int y, int x)`
- `void reset ()`
- `int sum ()`
- `vec< T > getvec (int y) const`
- `vec< T > getcol (int x) const`
- `T minpos (int &ypos, int &xpos)`
- `T maxpos (int &ypos, int &xpos)`
- `void CopyTo (T *pData)`
- `void CopyFrom (T *pData)`
- `template<class T2 >`
`void ConvertTo (T2 *pData)`
- `template<class T2 >`
`void ConvertFrom (T2 *pData)`
- `template<>`
`mtr< double > & operator<<= (const int Shl)`

Открытые атрибуты

- `int m_border`
- `int m_stride`
- `int m_height`
- `int m_width`
- `int m_size`
- `int m_x0`
- `int m_y0`
- `T * m_data`

Защищенные данные

- `T * m_container`

7.22.1 Подробное описание

```
template<class T>
class mtr< T >
```

См. определение в файле `tmatrix.h` строка 87

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tmatrix.h`

7.23 Структура nm16sc

Открытые атрибуты

- `signed short r`
- `signed short c`

7.23.1 Подробное описание

См. определение в файле nmtype.h строка 1243

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmtype.h

7.24 Класс nmchar

Открытые члены

- nmchar (unsigned int *p, int offset)
- nmchar (nmchar &ch)
- nmchar & operator= (nmchar ch)
- unsigned int operator+ (nmchar &ch)
- nmchar & operator&= (unsigned int val)
- nmchar & operator|= (unsigned int val)
- nmchar & operator= (unsigned int val)
- operator unsigned char ()
- uint8ptr operator& ()

Открытые атрибуты

- unsigned int * adr
- int idx

7.24.1 Подробное описание

См. определение в файле nmchar.h строка 13

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmchar.h

7.25 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]

7.25.1 Подробное описание

```
template<int N>
class nmchar1D< N >
```

См. определение в файле nmchar.h строка 363

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmchar.h

7.26 Шаблон класса nmchar2D< Y, X >

Открытые члены

- [uint8ptr](#) & operator[] (int idx)
- unsigned int * ptr ()

Открытые атрибуты

- [uint8ptr](#) arr
- unsigned int data [Y *X/4]

7.26.1 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

7.27 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

7.27.1 Подробное описание

```
template<class T>
class nmintpack< T >
```

См. определение в файле tnmvespack.h строка 28

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmtl/tnmvespack.h

7.28 Шаблон класса nmmtr< T >

```
#include <tnmmtr.h>
```

Открытые члены

- nmmtr (int Height, int Width, int Border=0)
- nmvec< T > operator[] (int y) const
- nmmtr (nmmtr< T > &mtr)
- nmmtr (T *Data, int Height, int Width, int Stride=0)
- nmmtr (const T *Data, int Height, int Width, int Stride=0)
- nmmtr< T > & operator= (const nmmtr< T > &mtr)
- nmmtr< T > & operator*= (const nmint< T > &val)
- template<class T2 >
nmmtr< T2 > operator* (const nmint< T2 > &val)
- template<class T2 >
nmvec< T2 > operator* (const nmvec< T2 > &vec)
- template<class T2 >
nmmtr< T2 > operator* (const nmmtr< T2 > &mtr)
- nmvec< T > & operator+= (const nmint< T > &val)
- nmmtr< T > & operator+= (const nmmtr< T > &mtr)
- nmmtr< T > operator+ (const nmmtr< T > &mtr) const
- nmmtr< T > & operator-= (const nmmtr< T > &mtr)
- nmmtr< T > operator- (const nmmtr< T > &mtr) const
- nmmtr< T > operator- () const
- nmmtr< T > & operator/= (const T val)
- nmmtr< T > operator/ (const nmint< T > val) const
- nmmtr< T > & operator>>= (const int shr)
- nmmtr< T > operator>> (const int shr) const
- nmmtr< T > & operator<<= (const int shl)
- nmmtr< T > operator<< (const int shl) const
- nmmtr< T > & operator|= (const nmmtr< T > &mtr)
- nmmtr< T > operator| (const nmmtr< T > &mtr) const
- nmmtr< T > & operator &= (const nmmtr< T > &mtr)

- `nmmtr< T > operator & (const nmmtr< T > &mtr) const`
- `nmmtr< T > & operator^= (const nmint< T > &val)`
- `nmmtr< T > operator^ (const nmint< T > &val) const`
- `nmmtr< T > operator~ () const`
- `T * addr (int y, int x)`
- `nmvec< T > vec (int y)`
- `void fill (nmint< T > &nVal)`
- `nmmtr< T > transpose ()`
- `template<class T2 >`
`void set (nmmtr< T2 > &mSrcMtr) const`
- `template<class T2 >`
`void set (mtr< T2 > &Mtr)`
- `void set (const T val)`
- `void reset ()`

Открытые атрибуты

- `int m_height`
- `int m_width`
- `int m_size`
- `int m_stride`
- `int m_border`
- `T * m_data`

Защищенные данные

- `T * m_container`

7.28.1 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`

7.29 Шаблон класса `nmmtrpack< T >`

```
#include <tnmmtrpack.h>
```


Открытые члены

- `nmmtrpack` (int Height, int Width, int Border=0)
- `__INLINE__ nmvecpack< T > operator[]` (int y) const
- `nmmtrpack` (const `nmmtrpack< T > &mtr`)
- `nmmtrpack` (T *Data, int Height, int Width, int Stride=0)
- `nmmtrpack` (const T *Data, int Height, int Width, int Stride=0)
- `nmmtrpack< T > & operator=` (const `nmmtrpack< T > &mtr`)
- `nmmtrpack< T > & operator*=` (const `nmint< T > &val`)
- `template<class T2 >`
`nmmtrpack< T2 > operator*` (const `nmint< T2 > &val`)
- `template<class T2 >`
`nmvec< T2 > operator*` (const `nmvec< T2 > &vec`)
- `template<class T2 >`
`nmmtrpack< T2 > operator*` (const `nmmtrpack< T2 > &mtr`)
- `nmvec< T > & operator+=` (const `nmint< T > &val`)
- `nmmtrpack< T > & operator+=` (const `nmmtrpack< T > &mtr`)
- `nmmtrpack< T > operator+` (const `nmmtrpack< T > &mtr`) const
- `nmmtrpack< T > & operator-=` (const `nmmtrpack< T > &mtr`)
- `nmmtrpack< T > operator-` (const `nmmtrpack< T > &mtr`) const
- `nmmtrpack< T > operator-` () const
- `nmmtrpack< T > & operator/=` (const T val)
- `nmmtrpack< T > operator/` (const `nmint< T > val`) const
- `nmmtrpack< T > & operator>>=` (const int shr)
- `nmmtrpack< T > operator>>` (const int shr) const
- `nmmtrpack< T > & operator<<=` (const int shl)
- `nmmtrpack< T > operator<<` (const int shl) const
- `nmmtrpack< T > & operator|=` (const `nmmtrpack< T > &mtr`)
- `nmmtrpack< T > operator|` (const `nmmtrpack< T > &mtr`) const
- `nmmtrpack< T > & operator &=` (const `nmmtrpack< T > &mtr`)
- `nmmtrpack< T > operator &` (const `nmmtrpack< T > &mtr`) const
- `nmmtrpack< T > & operator^=` (const `nmint< T > &val`)
- `nmmtrpack< T > operator^` (const `nmint< T > &val`) const
- `nmmtrpack< T > operator~` () const
- `__INLINE__ nmvec< T > GetVec` (int y)
- `__INLINE__ nmvec< T > GetVec` (int y, int x, int len)
- `__INLINE__ nmmtrpack< T > & SetMtr` (int y, int x, `nmmtrpack< T > &mSrc`)
- `__INLINE__ nmmtrpack< T > & GetMtr` (int y, int x, `nmmtrpack< T > &mRes`)
- `__INLINE__ nmmtrpack< T > GetMtr` (int y, int x, int height, int width)
- `__INLINE__ T * Addr` (int y, int x)
- `template<class T2 >`
`void Set` (`nmmtrpack< T2 > &mSrcMtr`) const
- `void InitConst` (`nmint< T > &nVal`)
- `nmmtrpack< T > transpose` ()
- `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`
- `bool m_flag_new`

7.29.1 Подробное описание

```
template<class T>
class nmmtrpack< T >
```

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. класс матриц.

См. определение в файле `tnmmtrpack.h` строка 37

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tnmmtrpack.h`

7.30 Структура `NmppiFFTSpec_32fc`

Открытые атрибуты

- `NmppsFFTSpec_32fc * specX`
- `NmppsFFTSpec_32fc * specY`

7.30.1 Подробное описание

См. определение в файле `fft_32fc.h` строка 21

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/fft_32fc.h`

7.31 Структура `NmppsFFTSpec`

Открытые атрибуты

- `nm32sc * buffer [FFT_SPEC_NUM_BUFFERS]`
- `void * fftTable [FFT_SPEC_NUM_TABLES]`
- `int shift [FFT_SPEC_NUM_SHIFTS]`
- `int amp [FFT_SPEC_NUM_AMPLITUDES]`
- `Free32Func * free`

7.31.1 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`

7.32 Структура NmppsFFTSpec_32fc

Открытые атрибуты

- `nm32fcr` * Buffers [NUMBUFF1]
- `nm32fcr` * Buffs [NUMBUFF2]
- `int` order

7.32.1 Подробное описание

См. определение в файле `fft_32fc.h` строка 14

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/fft_32fc.h`

7.33 Структура NmppsFrame_16s

Открытые атрибуты

- `void` * pull
- `nm16s` * data

7.33.1 Подробное описание

См. определение в файле `malloc32.h` строка 92

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/malloc32.h`

7.34 Структура NmppsFrame_16u

Открытые атрибуты

- `void` * pull
- `nm16u` * data

7.34.1 Подробное описание

См. определение в файле malloc32.h строка 87

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.35 Структура NmppsFrame_32s

Открытые атрибуты

- void * pull
- [nm32s](#) * data

7.35.1 Подробное описание

См. определение в файле malloc32.h строка 102

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.36 Структура NmppsFrame_32u

Открытые атрибуты

- void * pull
- [nm32u](#) * data

7.36.1 Подробное описание

См. определение в файле malloc32.h строка 97

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.37 Структура NmppsFrame_64s

Открытые атрибуты

- void * pull
- [nm64s](#) * data

7.37.1 Подробное описание

См. определение в файле malloc32.h строка 112

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.38 Структура NmppsFrame_64u

Открытые атрибуты

- void * pull
- nm64u * data

7.38.1 Подробное описание

См. определение в файле malloc32.h строка 107

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.39 Структура NmppsFrame_8s

Открытые атрибуты

- void * pull
- nm8s * data

7.39.1 Подробное описание

См. определение в файле malloc32.h строка 82

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.40 Структура NmppsFrame_8u

Открытые атрибуты

- void * pull
- nm8u * data

7.40.1 Подробное описание

См. определение в файле malloc32.h строка 77

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.41 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

7.41.1 Подробное описание

См. определение в файле malloc32.h строка 37

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.42 Структура NmppsTmpSpec

Открытые атрибуты

- void * buffer0
- void * buffer1

7.42.1 Подробное описание

См. определение в файле malloc32.h строка 55

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

7.43 Структура nmreg

```
#include <nmtype.h>
```

Открытые атрибуты

- int nVal

7.43.1 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](#)

7.44 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

7.44.1 Подробное описание

См. определение в файле nmshort.h строка 8

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/[nmshort.h](#)

7.45 Шаблон класса nmshort2D< Y, X >

Открытые члены

- `uint16ptr` & operator[] (int idx)
- `unsigned int * ptr` ()

Открытые атрибуты

- `uint16ptr` arr
- `unsigned int data [Y *X/2]`

7.45.1 Подробное описание

```
template<int Y, int X>
class nmshort2D< Y, X >
```

См. определение в файле nmshort.h строка 292

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmshort.h`

7.46 Шаблон класса nmvecpack< T >

```
#include <tnmvecpack.h>
```

Открытые члены

- `nmvecpack` (void *Data, int Size, int Border=0)
- `nmvecpack` (int Size, int Border=0)
- `nmvecpack` (const `nmvecpack`< T > &vec)
- `nmvecpack`< T > & operator= (const `nmvecpack`< T > &vec)
- `__INLINE__ nmintpack`< T > operator[] (int idx)
- `template<class T2 >`
`nmvecpack`< T2 > & operator*= (const `nmint`< T2 > val)
- `template<class T2 >`
`nmvecpack`< T2 > operator* (const `nmint`< T2 > &val) const
- `template<class T2 >`
`nmint`< T2 > operator* (const `nmvecpack`< T2 > &vec) const
- `nmvecpack`< T > & operator+= (const `nmint`< T > &val)
- `nmvecpack`< T > & operator+= (const `nmvecpack`< T > &vec)
- `nmvecpack`< T > operator+ (const `nmint`< T > &val) const
- `nmvecpack`< T > operator+ (const `nmvecpack`< T > &vec) const
- `nmvecpack`< T > & operator-= (const `nmint`< T > &val)
- `nmvecpack`< T > & operator-= (const `nmvecpack`< T > &vec)
- `nmvecpack`< T > operator- () const
- `nmvecpack`< T > operator- (const `nmint`< T > &val) const

- `nmvecpack< T > operator- (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator/= (const nmint< T > val)`
- `nmvecpack< T > operator/ (const T val) const`
- `nmvecpack< T > & operator>>= (const int shr)`
- `nmvecpack< T > operator>> (const int shr) const`
- `nmvecpack< T > & operator<<= (const int shl)`
- `nmvecpack< T > operator<< (const int shl) const`
- `nmvecpack< T > & operator|= (const nmint< T > &val)`
- `nmvecpack< T > & operator|= (const nmvecpack< T > &vec)`
- `nmvecpack< T > operator| (const nmint< T > &val) const`
- `nmvecpack< T > operator| (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator &= (const nmint< T > &val)`
- `nmvecpack< T > & operator &= (const nmvecpack< T > &vec)`
- `nmvecpack< T > operator & (const nmint< T > &val) const`
- `nmvecpack< T > operator & (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator^= (const nmint< T > &val)`
- `nmvecpack< T > & operator^= (const nmvecpack< T > &vec)`
- `nmvecpack< T > operator^ (const nmint< T > &val) const`
- `nmvecpack< T > operator^ (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator~ () const`
- `bool operator== (const nmvecpack< T > &vec) const`
- `bool operator!= (const nmvecpack< T > &vec) const`
- `template<class T2 >`
`void SetData (T2 *Data)`
- `template<class T2 >`
`void GetData (T2 *Data)`
- `void reset ()`

Открытые атрибуты

- `int m_size`
- `T * m_data`

Защищенные данные

- `T * m_container`
- `int m_border`

7.46.1 Подробное описание

```
template<class T>
class nmvecpack< T >
```

Класс векторов.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};
int Res [10];
nmvecpack<int> A0(3);
nmvecpack<int> B0(3);
nmvecpack<int> C0(3);
```

См. определение в файле `tnmvecpack.h` строка 137

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tnmvecpack.h`

7.47 Структура RGB24_nm8u

Открытые атрибуты

- nm8u nB
- nm8u nG
- nm8u nR

7.47.1 Подробное описание

См. определение в файле iDef.h строка 24

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

7.48 Структура RGB32_nm10s

Открытые атрибуты

- int nB:10
- int nG:10
- int nR:10
- int nA:2

7.48.1 Подробное описание

См. определение в файле iDef.h строка 56

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

7.49 Структура RGB32_nm10u

Открытые атрибуты

- unsigned int nB:10
- unsigned int nG:10
- unsigned int nR:10
- unsigned int nA:2

7.49.1 Подробное описание

См. определение в файле iDef.h строка 48

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

7.50 Структура RGB32_nm8s

Открытые атрибуты

- int nB:8
- int nG:8
- int nR:8
- int nA:8

7.50.1 Подробное описание

См. определение в файле iDef.h строка 40

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

7.51 Структура RGB32_nm8u

Открытые атрибуты

- unsigned int nB:8
- unsigned int nG:8
- unsigned int nR:8
- unsigned int nA:8

7.51.1 Подробное описание

См. определение в файле iDef.h строка 32

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

7.52 Структура RGB64_nm16u

Открытые атрибуты

- unsigned int nB:16
- unsigned int nG:16
- unsigned int nR:16
- unsigned int nA:16

7.52.1 Подробное описание

См. определение в файле iDef.h строка 64

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

7.53 Класс RPoint

Открытые члены

- RPoint (const [RPoint](#) &p)
- RPoint (double _x, double _y)
- [RPoint](#) & operator= (const [RPoint](#) &p)

Открытые атрибуты

- double x
- double y

7.53.1 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

7.54 Структура S_BufferInfo

класс буфер - заголовок в начале выделяемой динамической памяти

```
#include <multiheap.h>
```

Открытые члены

- `int * DataBegin ()`
Возвращает указатель на данные в буфере
- `int * DataEnd ()`
Возвращает указатель на конец данных в буфере (следует)
- `int * EndGuardBits ()`
Возвращает указатель на конечные защитные поля (2 слова)
- `unsigned CheckGuardBits ()`

Открытые атрибуты

- `int guardInfoBits0`
- `size_t size32Buffer`
< размер массива данных в буфере в 32-х словах
- `S_BufferInfo * pPrevBuffer`
< указатель на предыдущий буфер в списке
- `S_BufferInfo * pNextBuffer`
< указатель на следующий буфер в списке
- `bool isLocked`
< флаг блокировки буфера, запрещающий его удаление с помощью Release()
- `int guardInfoBits1`

7.54.1 Подробное описание

класс буфер - заголовок в начале выделяемой динамической памяти

См. определение в файле multiheap.h строка 14

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

7.55 Структура S_IMG_FilterKernel

Открытые атрибуты

- `nm32s * pDispArray`
- `nm32s * pWeightMatrix`

7.55.1 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

7.56 Структура S_IMG_FilterKernel_32s32s

Открытые атрибуты

- [nm32s](#) * pDispArray
- [nm32s](#) * pWeightMatrix
- int nKerWidth
- int nKerHeight

7.56.1 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

7.57 Структура s_int32x2

Открытые атрибуты

- int hi
- int lo

7.57.1 Подробное описание

См. определение в файле nmtype.h строка 258

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.58 Структура s_nm32fc

Открытые атрибуты

- float re
- float im

7.58.1 Подробное описание

См. определение в файле nmtype.h строка 1278

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.59 Структура s_nm32fcr

Открытые атрибуты

- float im
- float re

7.59.1 Подробное описание

См. определение в файле nmtype.h строка 1285

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.60 Структура s_nm32sc

Открытые атрибуты

- [nm32s](#) re
- [nm32s](#) im

7.60.1 Подробное описание

См. определение в файле nmtype.h строка 1267

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.61 Структура s_nm64sc

Открытые атрибуты

- long long [re](#)
- long long [im](#)

7.61.1 Подробное описание

См. определение в файле nmtype.h строка 1314

7.61.2 Данные класса

7.61.2.1 im

```
long long s_nm64sc::im
```

Мнимая часть комплексного числа.

См. определение в файле `nmtype.h` строка 1327

7.61.2.2 re

```
long long s_nm64sc::re
```

Вещественная часть комплексного числа.

См. определение в файле `nmtype.h` строка 1321

Объявления и описания членов структуры находятся в файле:

- [D:/GIT/nmpp/include/nmtype.h](#)

7.62 Структура `s_v16nm16s`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [4]`

7.62.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1026

Объявления и описания членов структуры находятся в файле:

- [D:/GIT/nmpp/include/nmtype.h](#)

7.63 Структура `s_v16nm16u`

```
#include <nmtype.h>
```


Открытые атрибуты

- unsigned long long vec [4]

7.63.1 Подробное описание

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

См. определение в файле nmtype.h строка 1164

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.64 Структура s_v16nm32s

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [8]

7.64.1 Подробное описание

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

См. определение в файле nmtype.h строка 1071

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.65 Структура s_v16nm32u

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [8]

7.65.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1208

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.66 Структура `s_v16nm4u`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

7.66.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1098

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.67 Структура `s_v16nm8s`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [2]`

7.67.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 991

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.68 Структура s_v16nm8u

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [2]

7.68.1 Подробное описание

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

См. определение в файле nmtypes.h строка 1131

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.69 Структура s_v2nm32s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

7.69.1 Подробное описание

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

См. определение в файле nmtypes.h строка 1038

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.70 Структура s_v2nm32u

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

7.70.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1175

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.71 Структура `s_v4nm16s`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

7.71.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1002

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.72 Структура `s_v4nm16u`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

7.72.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1142

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.73 Структура s_v4nm32s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [2]

7.73.1 Подробное описание

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

См. определение в файле nmtypes.h строка 1049

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.74 Структура s_v4nm32u

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [2]

7.74.1 Подробное описание

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

См. определение в файле nmtypes.h строка 1186

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.75 Структура s_v4nm8u

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

7.75.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1109

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.76 Структура `s_v8nm16s`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [2]`

7.76.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1015

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.77 Структура `s_v8nm16u`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [2]`

7.77.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1153

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.78 Структура s_v8nm32s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [4]

7.78.1 Подробное описание

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

См. определение в файле nmtypes.h строка 1060

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.79 Структура s_v8nm32u

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [4]

7.79.1 Подробное описание

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

См. определение в файле nmtypes.h строка 1197

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.80 Структура s_v8nm8s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

7.80.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 980

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.81 Структура `s_v8nm8u`

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

7.81.1 Подробное описание

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

См. определение в файле `nmtype.h` строка 1120

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

7.82 Структура `SpecTmp1`

Открытые атрибуты

- `void * buffer`
- `int status`
- `int mode`
- `fseq64 route`

7.82.1 Подробное описание

См. определение в файле `malloc32.h` строка 148

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/malloc32.h`

7.83 Структура `spot_struct`

Открытые атрибуты

- `int Xmin`
координаты минимального прямоугольника, содержащего пятно
- `int Ymin`
координаты минимального прямоугольника, содержащего пятно
- `int Xmax`
координаты минимального прямоугольника, содержащего пятно
- `int Ymax`
координаты минимального прямоугольника, содержащего пятно
- `int noPxl`
номер начального пиксела следующего пятна в массиве `pixels`.
- `int dtSpot`
время обработки пятна (в тактах процессора)

7.83.1 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`

7.84 Структура `tagSegmentInfo`

Открытые атрибуты

- `int xMin`
- `int yMin`
- `int xMax`
- `int yMax`
- `int N`

7.84.1 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`

7.85 Шаблон класса `tfixpoint< T, point >`

Открытые члены

- `INLINE tfixpoint (const int val)`
- `INLINE tfixpoint (const float val)`
- `INLINE tfixpoint (const double val)`
- `INLINE tfixpoint< T, point > & operator= (const int val)`
- `INLINE tfixpoint< T, point > & operator= (const float val)`
- `tfixpoint< T, point > & operator= (const double val)`
- `INLINE tfixpoint< T, point > & operator= (const tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > & operator= (const tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > & operator+= (const tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > operator- (const tfixpoint< T, point > &val) const`
- `INLINE tfixpoint< T, point > operator+ (const tfixpoint< T, point > &val) const`
- `INLINE tfixpoint< T, point > & operator++ (int)`
- `INLINE tfixpoint< T, point > & operator-- (int)`
- `template<class T2 , int point2>`
`INLINE tfixpoint< T, point > & operator*= (const tfixpoint< T2, point2 > val)`
- `INLINE tfixpoint< T, point > & operator*= (const int val)`
- `INLINE tfixpoint< T, point > & operator*= (const float val)`
- `INLINE tfixpoint< T, point > & operator*= (const double val)`
- `template<int point2>`
`INLINE tfixpoint< T, point > operator* (const tfixpoint< T, point2 > val) const`
- `INLINE tfixpoint< T, point > operator* (const int val) const`
- `INLINE tfixpoint< T, point > & operator/= (const tfixpoint< T, point > val)`
- `INLINE tfixpoint< T, point > operator/ (const tfixpoint< T, point > val) const`
- `INLINE tfixpoint< T, point > & operator>>= (const int y)`
- `INLINE tfixpoint< T, point > operator>> (const int y) const`
- `INLINE tfixpoint< T, point > & operator<<= (const int y)`
- `INLINE tfixpoint< T, point > operator<< (const int n) const`
- `INLINE tfixpoint< T, point > & operator^= (tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > operator^ (tfixpoint< T, point > &val) const`
- `INLINE tfixpoint< T, point > operator- () const`
- `bool operator> (const tfixpoint< T, point > &y) const`
- `bool operator>= (const tfixpoint< T, point > &y) const`
- `bool operator< (const tfixpoint< T, point > &y) const`
- `bool operator<= (const tfixpoint< T, point > &y) const`
- `bool operator== (const tfixpoint< T, point > &y) const`
- `bool operator!= (const tfixpoint< T, point > &y) const`
- `float fit ()`

Открытые атрибуты

- `T m_value`

7.85.1 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`

7.86 Структура Tmp2BuffSpec

Открытые атрибуты

- void * buffer0
- void * buffer1
- fseq64 route
- int mode
- int status

7.86.1 Подробное описание

См. определение в файле nmtype.h строка 1353

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

7.87 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

7.87.1 Подробное описание

См. определение в файле `nmshort.h` строка 57

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmshort.h`

7.88 Класс `uint8ptr`

Открытые члены

- `unsigned char * x86addr ()`
- `uint8ptr (void *p)`
- `uint8ptr (unsigned char *p)`
- `uint8ptr (const unsigned char *p)`
- `uint8ptr (char *p)`
- `uint8ptr (const uint8ptr &p)`
- `uint8ptr (unsigned int *p)`
- `uint8ptr (unsigned int *p, int offset)`
- `nmchar & operator[] (int idx)`
- `uint8ptr & operator= (unsigned int *ptr)`
- `bool operator< (uint8ptr ptr)`
- `bool operator> (uint8ptr ptr)`
- `bool operator>= (uint8ptr ptr)`
- `int operator- (uint8ptr ptr)`
- `uint8ptr & operator= (const uint8ptr &p)`
- `unsigned int * ptr ()`
- `nmchar & operator* ()`
- `uint8ptr operator+ (int idx)`
- `uint8ptr operator- (int idx)`
- `uint8ptr & operator+= (int idx)`
- `uint8ptr & operator-= (int idx)`
- `uint8ptr operator++ (int)`
- `uint8ptr & operator++ ()`
- `bool operator== (uint8ptr ptr)`

Открытые атрибуты

- `unsigned int * addr`
- `int indx`
- `nmchar arref`

7.88.1 Подробное описание

См. определение в файле `nmchar.h` строка 79

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmchar.h`

7.89 Структура v16nm4s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

7.89.1 Подробное описание

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

См. определение в файле nmtypes.h строка 957

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.90 Структура v4nm8s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

7.90.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 8р. чисел со знаком.

См. определение в файле nmtypes.h строка 969

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

7.91 Шаблон класса vec< T >

```
#include <tvector.h>
```

Открытые члены

- void resize (int Size, int Border=0)
- vec (T *Data, int Size, int Border=0)
- vec (int Size, int Border=0)
- vec (const [vec](#)< T > &vect)
- void reset ()
- [vec](#)< T > & InitRamp (T StartValue, T Increment)
- int MaxPos (T &maxval)
- int MinPos (T &minval)
- double Mean ()
- T & CustomMax ()
- [vec](#)< T > & operator= (const T &val)
- [vec](#)< T > & operator= (const [vec](#)< T > &vect)
- T * addr (int idx)
- T & operator[] (size_t idx)
- [vec](#)< T > & operator*= (const T &val)
- [vec](#)< T > operator* (const T &val) const
- T operator* (const [vec](#)< T > &vect) const
- [vec](#)< T > operator* (const [mtr](#)< T > matr) const
- [vec](#)< T > & operator+= (const T &val)
- [vec](#)< T > & operator+= (const [vec](#)< T > &vect)
- [vec](#)< T > operator+ (const T &val) const
- [vec](#)< T > operator+ (const [vec](#)< T > &vect) const
- [vec](#)< T > & operator-= (const T &val)
- [vec](#)< T > & operator-= (const [vec](#)< T > &vect)
- [vec](#)< T > operator- (const [vec](#)< T > &vect) const
- [vec](#)< T > operator- () const
- [vec](#)< T > & operator/= (const T val)
- [vec](#)< T > operator/ (const T val) const
- [vec](#)< T > & operator>>= (const int shr)
- [vec](#)< T > operator>> (const int shr) const
- [vec](#)< T > & operator<<= (const int shl)
- [vec](#)< T > operator<< (const int shl) const
- [vec](#)< T > & operator &= (const T &val)
- [vec](#)< T > & operator &= (const [vec](#)< T > &vect)
- [vec](#)< T > operator & (const T &val) const
- [vec](#)< T > operator & (const [vec](#)< T > &vect) const
- [vec](#)< T > & operator|= (const T &val)
- [vec](#)< T > & operator|= (const [vec](#)< T > &vect)
- [vec](#)< T > operator| (const T &val) const
- [vec](#)< T > operator| (const [vec](#)< T > &vect) const
- [vec](#)< T > & operator^= (const T &val)
- [vec](#)< T > & operator^= (const [vec](#)< T > &vect)
- [vec](#)< T > operator^ (const T &val) const
- [vec](#)< T > operator^ (const [vec](#)< T > &vect) const
- int sum ()
- bool operator== (const [vec](#)< T > &vect)
- bool operator!= (const [vec](#)< T > &vect)

Открытые атрибуты

- int m_border
- int size
- T * m_data

Защищенные данные

- `T * m_container`

7.91.1 Подробное описание

```
template<class T>
class vec< T >
```

Класс векторов.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};
int Res [10];
vec<int> A0(3);
vec<int> B0(3);
vec<int> C0(3);
scalar<int> a0(2);
A0[0]=1;
A0[1]=2;
A0[2]=3;
B0.SetData(Test);
B0=A0;
a0=A0[1];

C0=A0+B0;
C0=A0*a0;
a0=A0*A0;
```

См. определение в файле `tvector.h` строка 79

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tvector.h`

Глава 8

Файлы

8.1 Файл D:/GIT/nmpp/include/nmtype.h

Классы

- struct [nmreg](#)
- struct [int15in16x4](#)
- struct [s_int32x2](#)
- struct [int31in32x2](#)
- struct [int30in32x2](#)
- struct [v16nm4s](#)
- struct [v4nm8s](#)
- struct [s_v8nm8s](#)
- struct [s_v16nm8s](#)
- struct [s_v4nm16s](#)
- struct [s_v8nm16s](#)
- struct [s_v16nm16s](#)
- struct [s_v2nm32s](#)
- struct [s_v4nm32s](#)
- struct [s_v8nm32s](#)
- struct [s_v16nm32s](#)
- struct [s_v16nm4u](#)
- struct [s_v4nm8u](#)
- struct [s_v8nm8u](#)
- struct [s_v16nm8u](#)
- struct [s_v4nm16u](#)
- struct [s_v8nm16u](#)
- struct [s_v16nm16u](#)
- struct [s_v2nm32u](#)
- struct [s_v4nm32u](#)
- struct [s_v8nm32u](#)
- struct [s_v16nm32u](#)
- struct [nm16sc](#)
- struct [s_nm32sc](#)
- struct [s_nm32fc](#)
- struct [s_nm32fcr](#)
- struct [s_nm64sc](#)
- struct [Tmp2BuffSpec](#)

Макросы

- `#define MEM_LOCAL 0`
- `#define MEM_GLOBAL 1`
- `#define HEAP0 0`
- `#define HEAP1 1`
- `#define HEAP2 2`
- `#define HEAP3 3`
- `#define sizeof32(t) sizeof(t)`
- `#define VEC_NM1(X) unsigned data[(X)/32];`
- `#define VEC_NM2U(X) unsigned data[(X)/16];`
- `#define VEC_NM2S(X) int data[(X)/16];`
- `#define VEC_NM4U(X)`
- `#define VEC_NM4S(X) int data[(X)/8];`
- `#define VEC_NM8U(X) unsigned int data[(X)/4];`
- `#define VEC_NM8S(X)`
- `#define VEC_NM16U(X)`
- `#define VEC_NM16S(X)`
- `#define VEC_NM32U(X) nm32u data[(X)];`
- `#define VEC_NM32S(X)`
- `#define CAPACITY_nm64s 1`
- `#define CAPACITY_nm32s 2`
- `#define CAPACITY_nm16s 4`
- `#define CAPACITY_nm8s 8`
- `#define CAPACITY_nm4s 16`
- `#define CAPACITY_nm2s 32`
- `#define CAPACITY_nm1 64`
- `#define __INLINE__ static inline`

Определения типов

- `typedef signed long long INT64`
- `typedef unsigned long long UINT64`
- `typedef void nm1`
- `typedef void nm2s`
- `typedef void nm4s`
- `typedef void nm8s`
- `typedef signed char nm8s7b`
- `typedef void nm16s`
- `typedef nm16s nm16s15b`
- `typedef int nm32s`
- `typedef struct s_int32x2 int32x2`
- `typedef int nm32s31b`
- `typedef int nm32s30b`
- `typedef long long nm64s`
- `typedef nm64s nm64s63b`
- `typedef void nm2u`
- `typedef void nm4u`
- `typedef nm4u nm4u3b`
- `typedef unsigned char nm8u7b`
- `typedef void nm16u`
- `typedef nm16u nm16u15b`
- `typedef unsigned int nm32u`
- `typedef unsigned int nm32u31b`

- typedef unsigned long long [nm64u](#)
- typedef int [int1b](#)
- typedef int [int2b](#)
- typedef int [int3b](#)
- typedef int [int4b](#)
- typedef int [int7b](#)
- typedef int [int8b](#)
- typedef int [int15b](#)
- typedef int [int16b](#)
- typedef int [int30b](#)
- typedef int [int31b](#)
- typedef int [int32b](#)
- typedef INT64 [int63b](#)
- typedef INT64 [int64b](#)
- typedef unsigned int [uint1b](#)
- typedef unsigned int [uint2b](#)
- typedef unsigned int [uint3b](#)
- typedef unsigned int [uint4b](#)
- typedef unsigned int [uint7b](#)
- typedef unsigned int [uint8b](#)
- typedef unsigned int [uint15b](#)
- typedef unsigned int [uint16b](#)
- typedef unsigned int [uint31b](#)
- typedef unsigned int [uint32b](#)
- typedef UINT64 [uint63b](#)
- typedef [nm64u](#) [uint64b](#)
- typedef struct [s_v8nm8s](#) [v8nm8s](#)
- typedef struct [s_v16nm8s](#) [v16nm8s](#)
- typedef struct [s_v4nm16s](#) [v4nm16s](#)
- typedef struct [s_v8nm16s](#) [v8nm16s](#)
- typedef struct [s_v16nm16s](#) [v16nm16s](#)
- typedef struct [s_v2nm32s](#) [v2nm32s](#)
- typedef struct [s_v4nm32s](#) [v4nm32s](#)
- typedef struct [s_v8nm32s](#) [v8nm32s](#)
- typedef struct [s_v16nm32s](#) [v16nm32s](#)
- typedef [v16nm8s](#) [v16nm8s7b](#)
- typedef struct [s_v16nm4u](#) [v16nm4u](#)
- typedef struct [s_v4nm8u](#) [v4nm8u](#)
- typedef struct [s_v8nm8u](#) [v8nm8u](#)
- typedef struct [s_v16nm8u](#) [v16nm8u](#)
- typedef struct [s_v4nm16u](#) [v4nm16u](#)
- typedef struct [s_v8nm16u](#) [v8nm16u](#)
- typedef struct [s_v16nm16u](#) [v16nm16u](#)
- typedef struct [s_v2nm32u](#) [v2nm32u](#)
- typedef struct [s_v4nm32u](#) [v4nm32u](#)
- typedef struct [s_v8nm32u](#) [v8nm32u](#)
- typedef struct [s_v16nm32u](#) [v16nm32u](#)
- typedef [v16nm4u](#) [v16nm4b3u](#)
- typedef struct [s_nm32sc](#) [nm32sc](#)
- typedef struct [s_nm32fc](#) [nm32fc](#)
- typedef struct [s_nm32fcr](#) [nm32fcr](#)
- typedef float [nm32f](#)
- typedef double [nm64f](#)
- typedef struct [s_nm64sc](#) [nm64sc](#)
- typedef unsigned long long [uint64](#)

- typedef unsigned int uint32
- typedef uint64 fifo64
- typedef uint32 fseq32
- typedef uint64 seq64
- typedef uint64 fseq64

8.1.1 Подробное описание

Файл, содержащий определения типов упакованных данных.

8.1.2 Макросы

8.1.2.1 CAPACITY_nm64s

```
#define CAPACITY_nm64s 1
```

•

См. определение в файле nmtype.h строка 1379

8.1.2.2 VEC_NM16S

```
#define VEC_NM16S(  
    X )
```

Макроопределение:

```
int data[(X)/2]; \
void set(int i, int val){ ((unsigned short*)data)[i]=val;} \
short operator[] (int index){ return ((unsigned short*)data)[index];}
```

См. определение в файле nmtype.h строка 936

8.1.2.3 VEC_NM16U

```
#define VEC_NM16U(  
    X )
```

Макроопределение:

```
unsigned int data[(X)/2]; \
void set(int i, int val){ ((unsigned short*)data)[i]=val;} \
unsigned short operator[] (int index){ return ((unsigned short*)data)[index];}
```

См. определение в файле nmtype.h строка 931

8.1.2.4 VEC_NM32S

```
#define VEC_NM32S(  
    X )
```

Макроопределение:

```
int data[(X)]; \n  
void set(int i, int val){ ((int*)data)[i]=val;} \n  
int operator[] (int index){ return ((int*)data)[index];}
```

См. определение в файле nmtype.h строка 942

8.1.2.5 VEC_NM4U

```
#define VEC_NM4U(  
    X )
```

Макроопределение:

```
unsigned data[(X)/8]; \n  
uint4b get(int nIndex){return nmget((nm4u*)data,nIndex); } \n  
uint4b operator[] (int index){ return nmget((nm4u*)data,index);}
```

См. определение в файле nmtype.h строка 918

8.1.2.6 VEC_NM8S

```
#define VEC_NM8S(  
    X )
```

Макроопределение:

```
int data[(X)/4]; \n  
void set(int i, int val){ ((char*)data)[i]=val;} \n  
char operator[] (int index){ return ((char*)data)[index];}
```

См. определение в файле nmtype.h строка 926

Предметный указатель

- Арифметические действия, [106](#)
- Арифметические операции, [88](#), [196](#)
- Базовые регистровые функции библиотеки, [302](#)
- Быстрое преобразование Фурье, [159](#)
 - nmppsFFTFree_32fc, [159](#)
- Блочное переупорядочивание, [101](#)
- Целевые функции, [307](#)
- Элементарные функции, [305](#)
- Функции деинтерлейсинга, [89](#)
 - IMG_DeinterlaceBlend, [89](#)
 - IMG_DeinterlaceSplit, [89](#)
- Функции графического вывода текста, [110](#)
 - hex2ascii, [110](#)
 - IMG_Print8x15, [111](#)
- Функции обработки изображений, [299](#)
- Функции обработки сигналов, [298](#)
- Функции поддержки, [109](#), [126](#), [194](#)
- Инициализация, [84](#)
- Инициализация и копирование, [108](#), [112](#), [195](#)
- Изменение размеров, [158](#)
- КИХ-фильтрация, [91](#), [164](#)
- Логические и бинарные операции, [197](#)
- Масочная фильтрация, [107](#), [157](#)
- Матричные функции, [297](#)
- Операции сравнения, [198](#)
- Переупорядочивание и сортировка, [199](#)
- Переупорядочивание изображений, [100](#)
- Скалярные функции, [300](#)
- Свертка, [156](#)
- Типы данных, [295](#)
- Типы скалярных данных, [187](#)
 - int15b, [187](#)
 - int16b, [188](#)
 - int1b, [188](#)
 - int2b, [188](#)
 - int30b, [188](#)
 - int31b, [189](#)
 - int32b, [189](#)
 - int3b, [189](#)
 - int4b, [189](#)
 - int63b, [190](#)
 - int64b, [190](#)
 - int7b, [190](#)
 - int8b, [190](#)
 - uint15b, [191](#)
 - uint16b, [191](#)
 - uint1b, [191](#)
 - uint2b, [191](#)
 - uint31b, [192](#)
 - uint32b, [192](#)
 - uint3b, [192](#)
 - uint4b, [192](#)
 - uint63b, [193](#)
 - uint64b, [193](#)
 - uint7b, [193](#)
 - uint8b, [193](#)
- Типы векторных данных, [175](#)
 - nm1, [177](#)
 - nm16s, [177](#)
 - nm16s15b, [177](#)
 - nm16u, [178](#)
 - nm16u15b, [178](#)
 - nm2s, [178](#)
 - nm2u, [179](#)
 - nm32s, [179](#)
 - nm32s30b, [179](#)
 - nm32s31b, [180](#)
 - nm32u, [180](#)
 - nm32u31b, [180](#)
 - nm4s, [181](#)
 - nm4u, [181](#)
 - nm4u3b, [181](#)
 - nm64s, [182](#)
 - nm64s63b, [182](#)
 - nm64u, [182](#)
 - nm8s, [183](#)
 - nm8s7b, [183](#)
 - nm8u7b, [183](#)
 - v16nm16s, [184](#)
 - v16nm16u, [184](#)
 - v16nm32s, [184](#)
 - v16nm32u, [184](#)
 - v16nm4b3u, [184](#)
 - v16nm4u, [184](#)
 - v16nm8s, [185](#)
 - v16nm8s7b, [185](#)
 - v16nm8u, [185](#)
 - v2nm32s, [185](#)
 - v2nm32u, [185](#)
 - v4nm16s, [185](#)
 - v4nm16u, [185](#)
 - v4nm32s, [186](#)
 - v4nm32u, [186](#)
 - v4nm8u, [186](#)
 - v8nm16s, [186](#)
 - v8nm16u, [186](#)
 - v8nm32s, [186](#)
 - v8nm32u, [186](#)

- v8nm8s, [186](#)
 - v8nm8u, [186](#)
- Векторные функции, [296](#)
- Векторно-матричные операции, [127](#)
- функции взвешенного суммирования, [306](#)
- контроль переполнения, [303](#)
 - GetVec, [303](#)
 - operator<<, [304](#)
- AllocateMaxAvail
 - C_Heap, [379](#)
- C_2DSubPixelMinPosition, [375](#)
- C_2DTrigSubPixelMinPosition, [375](#)
- C_Allocator32, [376](#)
- C_BoxImg< T >, [377](#)
- C_BoxVec< T >, [377](#)
- C_Heap, [378](#)
 - AllocateMaxAvail, [379](#)
- C_Img< T >, [379](#)
- C_MultiHeap, [380](#)
- C_PlessyCornerDetector, [381](#)
- C_PlessyCornerDetector_16s, [382](#)
- C_PlessyCornerDetector_32f, [383](#)
- C_RingBuffer
 - PushRequest, [385](#)
- C_RingBuffer< T >, [384](#)
- C_RingBufferRemote< T >, [386](#)
- C_WarpImg< T >, [387](#)
- CAPACITY_nm64s
 - nmtype.h, [434](#)
- CIMG_FIR< nmbits_in, nmbits_out >, [387](#)
- CIMG_FIR
 - CIMG_FIR, [388](#)
 - Filter, [389](#)
 - SetWeights, [390](#)
- D:/GIT/nmpp/include/nmtype.h, [431](#)
- DFT-8, [17](#)
 - nmppsDFT8Fwd_32fc, [17](#)
- ds_struct, [390](#)
- FFT-1024, [31](#), [138](#)
 - FFT_Fwd1024, [138](#)
 - nmppsFFT1024Fwd_32fc, [31](#)
 - nmppsFFT1024FwdInitAlloc_32fc, [31](#)
- FFT-128, [25](#)
 - nmppsFFT128Fwd_32fc, [25](#)
 - nmppsFFT128FwdInitAlloc_32fc, [25](#)
- FFT-16, [19](#)
 - nmppsFFT16Fwd_32fc, [19](#)
 - nmppsFFT16FwdInitAlloc_32fc, [19](#)
- FFT-2048, [33](#), [143](#)
 - FFT_Fwd2048, [143](#)
 - nmppsFFT2048Fwd_32fc, [33](#)
 - nmppsFFT2048FwdInitAlloc_32fc, [33](#)
- FFT-256, [27](#), [128](#)
 - FFT_Fwd256, [128](#)
 - nmppsFFT256Fwd_32fc, [27](#)
 - nmppsFFT256FwdInitAlloc_32fc, [27](#)
- FFT-32, [21](#)
 - nmppsFFT32Fwd_32fc, [21](#)
 - nmppsFFT32FwdInitAlloc_32fc, [21](#)
- FFT-4096, [148](#)
 - FFT_Fwd4096, [148](#)
- FFT-512, [29](#), [133](#)
 - FFT_Fwd512, [133](#)
 - nmppsFFT512Fwd_32fc, [29](#)
 - nmppsFFT512FwdInitAlloc_32fc, [29](#)
- FFT-64, [23](#)
 - nmppsFFT64Fwd_32fc, [23](#)
 - nmppsFFT64FwdInitAlloc_32fc, [23](#)
- FFT-8192, [152](#)
 - FFT_Fwd8192, [152](#)
- FFT-Common, [52](#)
 - nmppsFFTFwd_32fc, [52](#)
 - nmppsFFTFwdInitAlloc_32fc, [52](#)
- FFT_Fwd1024
 - FFT-1024, [138](#)
- FFT_Fwd2048
 - FFT-2048, [143](#)
- FFT_Fwd256
 - FFT-256, [128](#)
- FFT_Fwd4096
 - FFT-4096, [148](#)
- FFT_Fwd512
 - FFT-512, [133](#)
- FFT_Fwd8192
 - FFT-8192, [152](#)
- FFT_Inv1024
 - IFFT-1024, [140](#)
- FFT_Inv2048
 - IFFT-2048, [145](#)
- FFT_Inv256
 - IFFT-256, [130](#)
- FFT_Inv4096
 - IFFT-4096, [150](#)
- FFT_Inv512
 - IFFT-512, [135](#)
- FFT_Inv8192
 - IFFT-8192, [154](#)
- Filter
 - CIMG_FIR, [389](#)
- Fix-point 32, [87](#)
- Fix-point 64, [86](#)
- FloodFill8
 - Floodfill, [92](#)
- Floodfill, [92](#)
 - FloodFill8, [92](#)
- GetVec
 - контроль переполнения, [303](#)
- hex2ascii
 - Функции графического вывода текста, [110](#)
- I_2DSubPixelMinPosition, [391](#)
- I_PlessyCornerDetector, [391](#)

- IDFT-8, [35](#)
 - nmppsDFT8Inv_32fc, [35](#)
- IFFT-1024, [48](#), [140](#)
 - FFT_Inv1024, [140](#)
 - nmppsFFT1024Inv_32fc, [48](#)
 - nmppsFFT1024InvInitAlloc_32fc, [48](#)
- IFFT-128, [42](#)
 - nmppsFFT128Inv_32fc, [42](#)
 - nmppsFFT128InvInitAlloc_32fc, [42](#)
- IFFT-16, [36](#)
 - nmppsFFT16Inv_32fc, [36](#)
 - nmppsFFT16InvInitAlloc_32fc, [36](#)
- IFFT-2048, [50](#), [145](#)
 - FFT_Inv2048, [145](#)
 - nmppsFFT2048Inv_32fc, [50](#)
 - nmppsFFT2048InvInitAlloc_32fc, [50](#)
- IFFT-256, [44](#), [130](#)
 - FFT_Inv256, [130](#)
 - nmppsFFT256Inv_32fc, [44](#)
 - nmppsFFT256InvInitAlloc_32fc, [44](#)
- IFFT-32, [38](#)
 - nmppsFFT32Inv_32fc, [38](#)
 - nmppsFFT32InvInitAlloc_32fc, [38](#)
- IFFT-4096, [150](#)
 - FFT_Inv4096, [150](#)
- IFFT-512, [46](#), [135](#)
 - FFT_Inv512, [135](#)
 - nmppsFFT512Inv_32fc, [46](#)
 - nmppsFFT512InvInitAlloc_32fc, [46](#)
- IFFT-64, [40](#)
 - nmppsFFT64Inv_32fc, [40](#)
 - nmppsFFT64InvInitAlloc_32fc, [40](#)
- IFFT-8192, [154](#)
 - FFT_Inv8192, [154](#)
- IFFT-Common, [55](#)
 - nmppsFFTInv_32fc, [55](#)
 - nmppsFFTInvInitAlloc_32fc, [55](#)
- IMG_Convert, [98](#)
- IMG_DeinterlaceBlend
 - Функции деинтерлейсинга, [89](#)
- IMG_DeinterlaceSplit
 - Функции деинтерлейсинга, [89](#)
- IMG_Free, [104](#)
- IMG_MergeFromBlocks, [103](#)
- IMG_Print8x15
 - Функции графического вывода текста, [111](#)
- IMG_RGB32ToGray, [99](#)
- IMG_Release, [105](#)
- IMG_SplitIntoBlocks, [102](#)
- im
 - s_nm64sc, [413](#)
- int15b
 - Типы скалярных данных, [187](#)
- int15in16x4, [392](#)
- int16b
 - Типы скалярных данных, [188](#)
- int1b
 - Типы скалярных данных, [188](#)
- int2b
 - Типы скалярных данных, [188](#)
- int30b
 - Типы скалярных данных, [188](#)
- int30in32x2, [392](#)
- int31b
 - Типы скалярных данных, [189](#)
- int31in32x2, [392](#)
- int32b
 - Типы скалярных данных, [189](#)
- int3b
 - Типы скалярных данных, [189](#)
- int4b
 - Типы скалярных данных, [189](#)
- int63b
 - Типы скалярных данных, [190](#)
- int64b
 - Типы скалярных данных, [190](#)
- int7b
 - Типы скалярных данных, [190](#)
- int8b
 - Типы скалярных данных, [190](#)
- Integer operations, [85](#)
- MTR_Addr, [123](#)
- MTR_Copy, [115](#)
- MTR_Copyau, [114](#)
- MTR_Free, [122](#)
- MTR_GetVal, [125](#)
- MTR_Malloc, [121](#)
- MTR_ProdUnitV, [120](#)
- MTR_SetVal, [124](#)
- mtr< T >, [393](#)
- nm1
 - Типы векторных данных, [177](#)
- nm16s
 - Типы векторных данных, [177](#)
- nm16s15b
 - Типы векторных данных, [177](#)
- nm16sc, [394](#)
- nm16u
 - Типы векторных данных, [178](#)
- nm16u15b
 - Типы векторных данных, [178](#)
- nm2s
 - Типы векторных данных, [178](#)
- nm2u
 - Типы векторных данных, [179](#)
- nm32s
 - Типы векторных данных, [179](#)
- nm32s30b
 - Типы векторных данных, [179](#)
- nm32s31b
 - Типы векторных данных, [180](#)
- nm32u
 - Типы векторных данных, [180](#)
- nm32u31b
 - Типы векторных данных, [180](#)

- nm4s
 - Типы векторных данных, [181](#)
- nm4u
 - Типы векторных данных, [181](#)
- nm4u3b
 - Типы векторных данных, [181](#)
- nm64s
 - Типы векторных данных, [182](#)
- nm64s63b
 - Типы векторных данных, [182](#)
- nm64u
 - Типы векторных данных, [182](#)
- nm8s
 - Типы векторных данных, [183](#)
- nm8s7b
 - Типы векторных данных, [183](#)
- nm8u7b
 - Типы векторных данных, [183](#)
- nmchar, [395](#)
- nmchar1D< N >, [395](#)
- nmchar2D< Y, X >, [396](#)
- nmintpack< T >, [396](#)
- nmmttr< T >, [397](#)
- nmmttrpack< T >, [398](#)
- nmppSplit, [293](#)
- nmppSplitEco, [294](#)
- nmppcDivC, [60](#)
- nmppcDoubleToFix32, [65](#)
- nmppcDoubleToFix64, [76](#)
- nmppcFix32ToDouble, [66](#)
- nmppcFix64Exp01, [81](#)
- nmppcFix64ToDouble, [77](#)
- nmppcFixArcTan32, [64](#)
- nmppcFixArcTan64, [80](#)
- nmppcFixDiv64, [78](#)
- nmppcFixDivMod32, [74](#)
- nmppcFixExp32, [62](#)
- nmppcFixInv32, [69](#)
- nmppcFixMul32, [68](#)
- nmppcFixSinCos32, [63](#)
- nmppcFixSinCos64, [79](#)
- nmppcFixSqrt32, [67](#)
- nmppcFixSqrt64, [75](#)
- nmppcProdC, [61](#)
- nmppcSqrt, [83](#)
- nmppcTblFixArcCos32, [71](#)
- nmppcTblFixArcSin32, [70](#)
- nmppcTblFixCos32, [72](#)
- nmppcTblFixSin32, [73](#)
- NmppiFFTSpec_32fc, [400](#)
- nmppmCopyua, [113](#)
- nmppmMul_mm, [116](#)
- nmppmMul_mv_, [118](#)
- nmppmMul_mv__AddC, [119](#)
- nmppsAbs, [200](#)
- nmppsAbs1, [201](#)
- nmppsAbsDiff, [209](#)
- nmppsAbsDiff1, [210](#)
- nmppsAdd, [204](#)
- nmppsAdd_AddC, [205](#)
- nmppsAddC, [203](#)
- nmppsAddr_, [286](#)
- nmppsAnd, [226](#)
- nmppsAnd4V_, [227](#)
- nmppsAndNotV_, [228](#)
- nmppsAndC, [225](#)
- nmppsClipCC_, [275](#)
- nmppsClipConvert_AddC_, [277](#)
- nmppsClipPowC_, [274](#)
- nmppsClipRShiftConvert_AddC_, [276](#)
- nmppsCmpEq0, [270](#)
 - nmppsCmpEq0_16u15b, [270](#)
 - nmppsCmpEq0_32u31b, [271](#)
 - nmppsCmpEq0_8u7b, [271](#)
- nmppsCmpEq0_16u15b
 - nmppsCmpEq0, [270](#)
- nmppsCmpEq0_32u31b
 - nmppsCmpEq0, [271](#)
- nmppsCmpEq0_8u7b
 - nmppsCmpEq0, [271](#)
- nmppsCmpEqV_, [284](#)
- nmppsCmpEqC, [279](#)
- nmppsCmpLt0, [269](#)
- nmppsCmpMinMaxV_, [272](#)
- nmppsCmpNe0, [280](#)
- nmppsCmpNeV_, [285](#)
- nmppsCmpNeC, [281](#)
- nmppsConvert, [245](#)
 - nmppsConvert_1s2s, [246](#)
 - nmppsConvert_1u2u, [246](#)
 - nmppsConvert_32s32fcr, [246](#)
 - nmppsConvert_32sc32fcr, [247](#)
 - nmppsConvertRisc_8u32u, [247](#)
- nmppsConvert_1s2s
 - nmppsConvert, [246](#)
- nmppsConvert_1u2u
 - nmppsConvert, [246](#)
- nmppsConvert_32s32fcr
 - nmppsConvert, [246](#)
- nmppsConvert_32sc32fcr
 - nmppsConvert, [247](#)
- nmppsConvertRisc_8u32u
 - nmppsConvert, [247](#)
- nmppsCopy_, [248](#)
- nmppsCopyua_, [249](#)
- nmppsDFT8Fwd_32fc
 - DFT-8, [17](#)
- nmppsDFT8Inv_32fc
 - IDFT-8, [35](#)
- nmppsDisplaceBits, [239](#)
- nmppsDivC, [218](#)
- nmppsDotProd, [221](#)
- nmppsFFT1024Fwd_32fc
 - FFT-1024, [31](#)
- nmppsFFT1024FwdInitAlloc_32fc
 - FFT-1024, [31](#)

- nmppsFFT1024Inv_32fc
IFFT-1024, [48](#)
- nmppsFFT1024InvInitAlloc_32fc
IFFT-1024, [48](#)
- nmppsFFT128Fwd_32fc
FFT-128, [25](#)
- nmppsFFT128FwdInitAlloc_32fc
FFT-128, [25](#)
- nmppsFFT128Inv_32fc
IFFT-128, [42](#)
- nmppsFFT128InvInitAlloc_32fc
IFFT-128, [42](#)
- nmppsFFT16Fwd_32fc
FFT-16, [19](#)
- nmppsFFT16FwdInitAlloc_32fc
FFT-16, [19](#)
- nmppsFFT16Inv_32fc
IFFT-16, [36](#)
- nmppsFFT16InvInitAlloc_32fc
IFFT-16, [36](#)
- nmppsFFT2048Fwd_32fc
FFT-2048, [33](#)
- nmppsFFT2048FwdInitAlloc_32fc
FFT-2048, [33](#)
- nmppsFFT2048Inv_32fc
IFFT-2048, [50](#)
- nmppsFFT2048InvInitAlloc_32fc
IFFT-2048, [50](#)
- nmppsFFT256Fwd_32fc
FFT-256, [27](#)
- nmppsFFT256FwdInitAlloc_32fc
FFT-256, [27](#)
- nmppsFFT256Inv_32fc
IFFT-256, [44](#)
- nmppsFFT256InvInitAlloc_32fc
IFFT-256, [44](#)
- nmppsFFT32Fwd_32fc
FFT-32, [21](#)
- nmppsFFT32FwdInitAlloc_32fc
FFT-32, [21](#)
- nmppsFFT32Inv_32fc
IFFT-32, [38](#)
- nmppsFFT32InvInitAlloc_32fc
IFFT-32, [38](#)
- nmppsFFT512Fwd_32fc
FFT-512, [29](#)
- nmppsFFT512FwdInitAlloc_32fc
FFT-512, [29](#)
- nmppsFFT512Inv_32fc
IFFT-512, [46](#)
- nmppsFFT512InvInitAlloc_32fc
IFFT-512, [46](#)
- nmppsFFT64Fwd_32fc
FFT-64, [23](#)
- nmppsFFT64FwdInitAlloc_32fc
FFT-64, [23](#)
- nmppsFFT64Inv_32fc
IFFT-64, [40](#)
- nmppsFFT64InvInitAlloc_32fc
IFFT-64, [40](#)
- nmppsFFTFree_32fc
Быстрое преобразование Фурье, [159](#)
- nmppsFFTFwd_32fc
FFT-Common, [52](#)
- nmppsFFTFwdInitAlloc_32fc
FFT-Common, [52](#)
- nmppsFFTInv_32fc
IFFT-Common, [55](#)
- nmppsFFTInvInitAlloc_32fc
IFFT-Common, [55](#)
- NmppsFFTSpec, [400](#)
- NmppsFFTSpec_32fc, [401](#)
- nmppsFIR_Xs, [165](#)
- nmppsFIRFree, [169](#)
- nmppsFIRGetStateSize_Xs, [168](#)
- nmppsFIRInit_Xs, [166](#)
- nmppsFIRInitAlloc_Xs, [167](#)
- nmppsFirstNonZeroIndx, [261](#)
- nmppsFirstZeroIndx, [260](#)
- NmppsFrame_16s, [401](#)
- NmppsFrame_16u, [401](#)
- NmppsFrame_32s, [402](#)
- NmppsFrame_32u, [402](#)
- NmppsFrame_64s, [402](#)
- NmppsFrame_64u, [403](#)
- NmppsFrame_8s, [403](#)
- NmppsFrame_8u, [403](#)
- nmppsFree, [59](#)
- nmppsGetVal_, [288](#)
- nmppsGetVal_(return), [289](#)
- nmppsLastNonZeroIndx, [263](#)
- nmppsLastZeroIndx, [262](#)
- nmppsMalloc, [58](#)
- NmppsMallocSpec, [404](#)
- nmppsMaskV_, [235](#)
- nmppsMax_, [251](#)
- nmppsMax_16s15b, [251](#)
- nmppsMax_32s31b, [252](#)
- nmppsMax_8s7b, [252](#)
- nmppsMax_16s15b
nmppsMax_, [251](#)
- nmppsMax_32s31b
nmppsMax_, [252](#)
- nmppsMax_8s7b
nmppsMax_, [252](#)
- nmppsMaxEvery_, [265](#)
- nmppsMaxIndx_, [255](#)
- nmppsMin, [253](#)
- nmppsMin_16s15b, [253](#)
- nmppsMin_32s31b, [254](#)
- nmppsMin_8s7b, [254](#)
- nmppsMin_16s15b
nmppsMin, [253](#)
- nmppsMin_32s31b
nmppsMin, [254](#)
- nmppsMin_8s7b

- nmppsMin, 254
- nmppsMinCmpLtV_, 267
- nmppsMinEvery_, 264
- nmppsMinIndx_, 257
- nmppsMinIndxVN_, 259
- nmppsMul_AddC, 212
- nmppsMulC_AddC_2x32s
 - nmppsMulC_AddC, 213
- nmppsMulC_AddV_AddC, 216
- nmppsMulC_AddC, 213
 - nmppsMulC_AddC_2x32s, 213
- nmppsMulC, 211
- nmppsNeg, 202
- nmppsNot_, 224
- nmppsOr, 230
- nmppsOr3V_, 231
- nmppsOr4V_, 232
- nmppsOrC, 229
- nmppsRShiftC_, 237
- nmppsRShiftC_AddC_, 238
- nmppsRShiftC_MulC_AddC, 215
- nmppsRShiftC, 236
- nmppsRamp_, 244
- nmppsRand, 82
- nmppsRandUniform, 242
 - nmppsRandUniform_64s, 242
- nmppsRandUniform_, 243
- nmppsRandUniform_64s
 - nmppsRandUniform, 242
- nmppsRemap_, 291
- nmppsSet-инициализация, 241
- nmppsSetVal_, 287
- nmppsSub, 208
- nmppsSubCRev, 207
- nmppsSubC, 206
- nmppsSum, 220
- nmppsSumN, 217
- nmppsSwap_, 250
- NmppsTmpSpec, 404
- nmppsWeightedSum, 223
- nmppsXor, 234
- nmppsXorC, 233
- nmreg, 405
- nmshort, 405
- nmshort2D< Y, X >, 406
- nmtype.h
 - CAPACITY_nm64s, 434
 - VEC_NM16S, 434
 - VEC_NM16U, 434
 - VEC_NM32S, 434
 - VEC_NM4U, 435
 - VEC_NM8S, 435
- nmvecpack< T >, 406
- operator<<
 - контроль переполнения, 304
- PushRequest
 - C_RingBuffer, 385
- RGB24_nm8u, 408
- RGB32_nm10s, 408
- RGB32_nm10u, 408
- RGB32_nm8s, 409
- RGB32_nm8u, 409
- RGB64_nm16u, 410
- RPoint, 410
- re
 - s_nm64sc, 414
- S_BufferInfo, 410
- S_IMG_FilterKernel, 411
- S_IMG_FilterKernel_32s32s, 412
- s_int32x2, 412
- s_nm32fc, 412
- s_nm32fcr, 413
- s_nm32sc, 413
- s_nm64sc, 413
 - im, 413
 - re, 414
- s_vl6nm16s, 414
- s_vl6nm16u, 414
- s_vl6nm32s, 415
- s_vl6nm32u, 415
- s_vl6nm4u, 416
- s_vl6nm8s, 416
- s_vl6nm8u, 417
- s_v2nm32s, 417
- s_v2nm32u, 417
- s_v4nm16s, 418
- s_v4nm16u, 418
- s_v4nm32s, 419
- s_v4nm32u, 419
- s_v4nm8u, 419
- s_v8nm16s, 420
- s_v8nm16u, 420
- s_v8nm32s, 421
- s_v8nm32u, 421
- s_v8nm8s, 421
- s_v8nm8u, 422
- SIG_CreateResample, 172
- SIG_Median3, 163
- SIG_Resample_perf, 174
- SIG_ResampleDown2, 170
- SIG_ResampleUp3Down2, 171
- SIG_SetResample, 173
- SIG_XCorr, 161
- SetWeights
 - CIMG_FIR, 390
- SpecTmp1, 422
- spot_struct, 423
- tagSegmentInfo, 423
- tfixpoint< T, point >, 424
- Tmp2BuffSpec, 425
- uint15b
 - Типы скалярных данных, 191
- uint16b

Типы скалярных данных, 191
 uint16ptr, 425
 uint1b
 Типы скалярных данных, 191
 uint2b
 Типы скалярных данных, 191
 uint31b
 Типы скалярных данных, 192
 uint32b
 Типы скалярных данных, 192
 uint3b
 Типы скалярных данных, 192
 uint4b
 Типы скалярных данных, 192
 uint63b
 Типы скалярных данных, 193
 uint64b
 Типы скалярных данных, 193
 uint7b
 Типы скалярных данных, 193
 uint8b
 Типы скалярных данных, 193
 uint8ptr, 426

 v16nm16s
 Типы векторных данных, 184
 v16nm16u
 Типы векторных данных, 184
 v16nm32s
 Типы векторных данных, 184
 v16nm32u
 Типы векторных данных, 184
 v16nm4b3u
 Типы векторных данных, 184
 v16nm4s, 427
 v16nm4u
 Типы векторных данных, 184
 v16nm8s
 Типы векторных данных, 185
 v16nm8s7b
 Типы векторных данных, 185
 v16nm8u
 Типы векторных данных, 185
 v2nm32s
 Типы векторных данных, 185
 v2nm32u
 Типы векторных данных, 185
 v4nm16s
 Типы векторных данных, 185
 v4nm16u
 Типы векторных данных, 185
 v4nm32s
 Типы векторных данных, 186
 v4nm32u
 Типы векторных данных, 186
 v4nm8s, 427
 v4nm8u
 Типы векторных данных, 186
 v8nm16s

Типы векторных данных, 186
 v8nm16u
 Типы векторных данных, 186
 v8nm32s
 Типы векторных данных, 186
 v8nm32u
 Типы векторных данных, 186
 v8nm8s
 Типы векторных данных, 186
 v8nm8u
 Типы векторных данных, 186
 VEC_NM16S
 nmtype.h, 434
 VEC_NM16U
 nmtype.h, 434
 VEC_NM32S
 nmtype.h, 434
 VEC_NM4U
 nmtype.h, 435
 VEC_NM8S
 nmtype.h, 435
 VEC_QSort, 290
 vec< T >, 427
 Vec_0_sub_data, 308
 Vec_Abs, 329
 Vec_AccMul1D1W32_AddVr, 373
 Vec_Add, 330
 Vec_Add_VV_shift, 313
 Vec_And, 325
 Vec_BuildDiagWeights16, 366
 Vec_BuildDiagWeights8, 365
 Vec_ClipExt, 331
 Vec_ClipMul2D2W8_AddVr, 332
 Vec_ClipMulNDNW2_AddVr, 333
 Vec_ClipMulNDNW4_AddVr, 334
 Vec_ClipMulNDNW8_AddVr, 335
 Vec_CompareMaxV, 362
 Vec_CompareMinV, 361
 Vec_DupValueInVector16, 364
 Vec_DupValueInVector8, 363
 Vec_FilterCoreRow2, 322
 Vec_FilterCoreRow4, 323
 Vec_FilterCoreRow8, 324
 Vec_IncNeg, 336
 Vec_MUL_2V4toW8_shift, 349
 Vec_MUL_2V8toW16_shift, 350
 Vec_Mask, 326
 Vec_MaxVal, 369
 Vec_MaxVal_v4nm16s, 368
 Vec_MaxVal_v8nm8s, 367
 Vec_MinVal, 372
 Vec_MinVal_v4nm16s, 371
 Vec_MinVal_v8nm8s, 370
 Vec_Mul2D2W1_AddVr, 337
 Vec_Mul2D2W2_AddVr, 338
 Vec_Mul2D2W4_AddVr, 339
 Vec_Mul2D2W8_AddVr, 340
 Vec_Mul3D3W2_AddVr, 341

Vec_Mul3D3W8_AddVr, [342](#)
Vec_Mul4D4W2_AddVr, [343](#)
Vec_MulVN_AddVN, [344](#)
Vec_Or, [327](#)
Vec_Sub, [345](#)
Vec_SubAbs, [346](#)
Vec_SubVN_Abs, [347](#)
Vec_Swap, [348](#)
Vec_Xor, [328](#)
Vec_activate_data, [309](#)
Vec_activate_data_add_0, [310](#)
Vec_activate_data_add_ram, [312](#)
Vec_activate_data_xor_data, [311](#)
Vec_afifo, [314](#)
Vec_data, [315](#)
Vec_data_add_afifo, [316](#)
Vec_data_add_ram, [317](#)
Vec_data_and_ram, [318](#)
Vec_data_or_ram, [319](#)
Vec_data_sub_ram, [320](#)
Vec_data_xor_ram, [321](#)
Vec_not_data, [351](#)
Vec_ram, [352](#)
Vec_ram_sub_data, [353](#)
Vec_vsum_activate_data_0, [354](#)
Vec_vsum_data_0, [355](#)
Vec_vsum_data_afifo, [356](#)
Vec_vsum_data_vr, [357](#)
Vec_vsum_shift_data_0, [358](#)
Vec_vsum_shift_data_afifo, [360](#)
Vec_vsum_shift_data_vr, [359](#)