

fovPath1118_ver2

1.0.0

Generated by Doxygen 1.8.20

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 coder Namespace Reference	9
5.1.1 Typedef Documentation	9
5.1.1.1 SizeType	9
5.2 coder::detail Namespace Reference	10
6 Class Documentation	11
6.1 coder::array< T, N > Class Template Reference	11
6.1.1 Detailed Description	15
6.1.2 Member Typedef Documentation	15
6.1.2.1 Base	15
6.1.2.2 size_type	15
6.1.2.3 value_type	15
6.1.3 Constructor & Destructor Documentation	15
6.1.3.1 array() [1/4]	16
6.1.3.2 array() [2/4]	16
6.1.3.3 array() [3/4]	16
6.1.3.4 array() [4/4]	16
6.1.4 Member Function Documentation	16
6.1.4.1 at() [1/20]	17
6.1.4.2 at() [2/20]	17
6.1.4.3 at() [3/20]	17
6.1.4.4 at() [4/20]	17
6.1.4.5 at() [5/20]	18
6.1.4.6 at() [6/20]	18
6.1.4.7 at() [7/20]	18
6.1.4.8 at() [8/20]	18
6.1.4.9 at() [9/20]	19
6.1.4.10 at() [10/20]	19
6.1.4.11 at() [11/20]	19
6.1.4.12 at() [12/20]	20
6.1.4.13 at() [13/20]	20

6.1.4.14 at() [14/20]	20
6.1.4.15 at() [15/20]	21
6.1.4.16 at() [16/20]	21
6.1.4.17 at() [17/20]	21
6.1.4.18 at() [18/20]	22
6.1.4.19 at() [19/20]	22
6.1.4.20 at() [20/20]	22
6.1.4.21 begin() [1/2]	23
6.1.4.22 begin() [2/2]	23
6.1.4.23 capacity()	23
6.1.4.24 clear()	23
6.1.4.25 data() [1/2]	23
6.1.4.26 data() [2/2]	24
6.1.4.27 end() [1/2]	24
6.1.4.28 end() [2/2]	24
6.1.4.29 ensureCapacity()	24
6.1.4.30 index() [1/10]	25
6.1.4.31 index() [2/10]	25
6.1.4.32 index() [3/10]	25
6.1.4.33 index() [4/10]	25
6.1.4.34 index() [5/10]	26
6.1.4.35 index() [6/10]	26
6.1.4.36 index() [7/10]	26
6.1.4.37 index() [8/10]	27
6.1.4.38 index() [9/10]	27
6.1.4.39 index() [10/10]	27
6.1.4.40 is_owner()	28
6.1.4.41 numel()	28
6.1.4.42 operator[]() [1/2]	28
6.1.4.43 operator[]() [2/2]	28
6.1.4.44 reshape() [1/10]	28
6.1.4.45 reshape() [2/10]	29
6.1.4.46 reshape() [3/10]	29
6.1.4.47 reshape() [4/10]	29
6.1.4.48 reshape() [5/10]	29
6.1.4.49 reshape() [6/10]	30
6.1.4.50 reshape() [7/10]	30
6.1.4.51 reshape() [8/10]	30
6.1.4.52 reshape() [9/10]	31
6.1.4.53 reshape() [10/10]	31
6.1.4.54 reshape_n()	31
6.1.4.55 set() [1/10]	32

6.1.4.56 <code>set()</code> [2/10]	32
6.1.4.57 <code>set()</code> [3/10]	32
6.1.4.58 <code>set()</code> [4/10]	32
6.1.4.59 <code>set()</code> [5/10]	33
6.1.4.60 <code>set()</code> [6/10]	33
6.1.4.61 <code>set()</code> [7/10]	33
6.1.4.62 <code>set()</code> [8/10]	34
6.1.4.63 <code>set()</code> [9/10]	34
6.1.4.64 <code>set()</code> [10/10]	35
6.1.4.65 <code>set_owner()</code>	35
6.1.4.66 <code>set_size()</code> [1/10]	35
6.1.4.67 <code>set_size()</code> [2/10]	36
6.1.4.68 <code>set_size()</code> [3/10]	36
6.1.4.69 <code>set_size()</code> [4/10]	36
6.1.4.70 <code>set_size()</code> [5/10]	37
6.1.4.71 <code>set_size()</code> [6/10]	37
6.1.4.72 <code>set_size()</code> [7/10]	37
6.1.4.73 <code>set_size()</code> [8/10]	38
6.1.4.74 <code>set_size()</code> [9/10]	38
6.1.4.75 <code>set_size()</code> [10/10]	39
6.1.4.76 <code>size()</code> [1/2]	39
6.1.4.77 <code>size()</code> [2/2]	39
6.1.5 Member Data Documentation	39
6.1.5.1 <code>data_</code>	40
6.1.5.2 <code>size_</code>	40
6.2 <code>coder::array<char_T, 2></code> Class Reference	40
6.2.1 Detailed Description	45
6.2.2 Member Typedef Documentation	45
6.2.2.1 <code>Base</code>	45
6.2.2.2 <code>size_type</code>	45
6.2.2.3 <code>value_type</code>	45
6.2.3 Constructor & Destructor Documentation	46
6.2.3.1 <code>array()</code> [1/6]	46
6.2.3.2 <code>array()</code> [2/6]	46
6.2.3.3 <code>array()</code> [3/6]	46
6.2.3.4 <code>array()</code> [4/6]	46
6.2.3.5 <code>array()</code> [5/6]	47
6.2.3.6 <code>array()</code> [6/6]	47
6.2.4 Member Function Documentation	47
6.2.4.1 <code>at()</code> [1/20]	48
6.2.4.2 <code>at()</code> [2/20]	48
6.2.4.3 <code>at()</code> [3/20]	48

6.2.4.4 <code>at()</code> [4/20]	48
6.2.4.5 <code>at()</code> [5/20]	49
6.2.4.6 <code>at()</code> [6/20]	49
6.2.4.7 <code>at()</code> [7/20]	49
6.2.4.8 <code>at()</code> [8/20]	49
6.2.4.9 <code>at()</code> [9/20]	50
6.2.4.10 <code>at()</code> [10/20]	50
6.2.4.11 <code>at()</code> [11/20]	50
6.2.4.12 <code>at()</code> [12/20]	51
6.2.4.13 <code>at()</code> [13/20]	51
6.2.4.14 <code>at()</code> [14/20]	51
6.2.4.15 <code>at()</code> [15/20]	52
6.2.4.16 <code>at()</code> [16/20]	52
6.2.4.17 <code>at()</code> [17/20]	52
6.2.4.18 <code>at()</code> [18/20]	53
6.2.4.19 <code>at()</code> [19/20]	53
6.2.4.20 <code>at()</code> [20/20]	53
6.2.4.21 <code>begin()</code> [1/2]	54
6.2.4.22 <code>begin()</code> [2/2]	54
6.2.4.23 <code>capacity()</code>	54
6.2.4.24 <code>clear()</code>	54
6.2.4.25 <code>data()</code> [1/2]	54
6.2.4.26 <code>data()</code> [2/2]	55
6.2.4.27 <code>end()</code> [1/2]	55
6.2.4.28 <code>end()</code> [2/2]	55
6.2.4.29 <code>ensureCapacity()</code>	55
6.2.4.30 <code>index()</code> [1/10]	56
6.2.4.31 <code>index()</code> [2/10]	56
6.2.4.32 <code>index()</code> [3/10]	56
6.2.4.33 <code>index()</code> [4/10]	56
6.2.4.34 <code>index()</code> [5/10]	57
6.2.4.35 <code>index()</code> [6/10]	57
6.2.4.36 <code>index()</code> [7/10]	57
6.2.4.37 <code>index()</code> [8/10]	58
6.2.4.38 <code>index()</code> [9/10]	58
6.2.4.39 <code>index()</code> [10/10]	58
6.2.4.40 <code>is_owner()</code>	59
6.2.4.41 <code>numel()</code>	59
6.2.4.42 <code>operator std::string()</code>	59
6.2.4.43 <code>operator=()</code> [1/2]	60
6.2.4.44 <code>operator=()</code> [2/2]	60
6.2.4.45 <code>operator[]()</code> [1/2]	61

6.2.4.46 operator[]() [2/2]	61
6.2.4.47 reshape() [1/10]	61
6.2.4.48 reshape() [2/10]	61
6.2.4.49 reshape() [3/10]	62
6.2.4.50 reshape() [4/10]	62
6.2.4.51 reshape() [5/10]	62
6.2.4.52 reshape() [6/10]	62
6.2.4.53 reshape() [7/10]	63
6.2.4.54 reshape() [8/10]	63
6.2.4.55 reshape() [9/10]	63
6.2.4.56 reshape() [10/10]	64
6.2.4.57 reshape_n()	64
6.2.4.58 set() [1/10]	64
6.2.4.59 set() [2/10]	64
6.2.4.60 set() [3/10]	65
6.2.4.61 set() [4/10]	65
6.2.4.62 set() [5/10]	65
6.2.4.63 set() [6/10]	66
6.2.4.64 set() [7/10]	66
6.2.4.65 set() [8/10]	66
6.2.4.66 set() [9/10]	67
6.2.4.67 set() [10/10]	67
6.2.4.68 set_owner()	68
6.2.4.69 set_size() [1/10]	68
6.2.4.70 set_size() [2/10]	68
6.2.4.71 set_size() [3/10]	69
6.2.4.72 set_size() [4/10]	69
6.2.4.73 set_size() [5/10]	69
6.2.4.74 set_size() [6/10]	70
6.2.4.75 set_size() [7/10]	70
6.2.4.76 set_size() [8/10]	70
6.2.4.77 set_size() [9/10]	71
6.2.4.78 set_size() [10/10]	71
6.2.4.79 size() [1/2]	72
6.2.4.80 size() [2/2]	72
6.2.5 Member Data Documentation	72
6.2.5.1 data_	72
6.2.5.2 size_	72
6.3 coder::array< T, 1 > Class Template Reference	73
6.3.1 Detailed Description	77
6.3.2 Member Typedef Documentation	77
6.3.2.1 Base	77

6.3.2.2 <code>size_type</code>	77
6.3.2.3 <code>value_type</code>	77
6.3.3 Constructor & Destructor Documentation	78
6.3.3.1 <code>array()</code> [1/4]	78
6.3.3.2 <code>array()</code> [2/4]	78
6.3.3.3 <code>array()</code> [3/4]	78
6.3.3.4 <code>array()</code> [4/4]	78
6.3.4 Member Function Documentation	79
6.3.4.1 <code>at()</code> [1/20]	79
6.3.4.2 <code>at()</code> [2/20]	79
6.3.4.3 <code>at()</code> [3/20]	79
6.3.4.4 <code>at()</code> [4/20]	79
6.3.4.5 <code>at()</code> [5/20]	80
6.3.4.6 <code>at()</code> [6/20]	80
6.3.4.7 <code>at()</code> [7/20]	80
6.3.4.8 <code>at()</code> [8/20]	80
6.3.4.9 <code>at()</code> [9/20]	81
6.3.4.10 <code>at()</code> [10/20]	81
6.3.4.11 <code>at()</code> [11/20]	81
6.3.4.12 <code>at()</code> [12/20]	82
6.3.4.13 <code>at()</code> [13/20]	82
6.3.4.14 <code>at()</code> [14/20]	82
6.3.4.15 <code>at()</code> [15/20]	83
6.3.4.16 <code>at()</code> [16/20]	83
6.3.4.17 <code>at()</code> [17/20]	83
6.3.4.18 <code>at()</code> [18/20]	84
6.3.4.19 <code>at()</code> [19/20]	84
6.3.4.20 <code>at()</code> [20/20]	84
6.3.4.21 <code>begin()</code> [1/2]	85
6.3.4.22 <code>begin()</code> [2/2]	85
6.3.4.23 <code>capacity()</code>	85
6.3.4.24 <code>clear()</code>	85
6.3.4.25 <code>data()</code> [1/2]	85
6.3.4.26 <code>data()</code> [2/2]	86
6.3.4.27 <code>end()</code> [1/2]	86
6.3.4.28 <code>end()</code> [2/2]	86
6.3.4.29 <code>ensureCapacity()</code>	86
6.3.4.30 <code>index()</code> [1/10]	87
6.3.4.31 <code>index()</code> [2/10]	87
6.3.4.32 <code>index()</code> [3/10]	87
6.3.4.33 <code>index()</code> [4/10]	87
6.3.4.34 <code>index()</code> [5/10]	88

6.3.4.35 index() [6/10]	88
6.3.4.36 index() [7/10]	88
6.3.4.37 index() [8/10]	89
6.3.4.38 index() [9/10]	89
6.3.4.39 index() [10/10]	89
6.3.4.40 is_owner()	90
6.3.4.41 numel()	90
6.3.4.42 operator std::vector< T >()	90
6.3.4.43 operator=()	91
6.3.4.44 operator[]() [1/2]	91
6.3.4.45 operator[]() [2/2]	91
6.3.4.46 reshape() [1/10]	92
6.3.4.47 reshape() [2/10]	92
6.3.4.48 reshape() [3/10]	92
6.3.4.49 reshape() [4/10]	92
6.3.4.50 reshape() [5/10]	93
6.3.4.51 reshape() [6/10]	93
6.3.4.52 reshape() [7/10]	93
6.3.4.53 reshape() [8/10]	94
6.3.4.54 reshape() [9/10]	94
6.3.4.55 reshape() [10/10]	94
6.3.4.56 reshape_n()	95
6.3.4.57 set() [1/10]	95
6.3.4.58 set() [2/10]	95
6.3.4.59 set() [3/10]	95
6.3.4.60 set() [4/10]	96
6.3.4.61 set() [5/10]	96
6.3.4.62 set() [6/10]	96
6.3.4.63 set() [7/10]	97
6.3.4.64 set() [8/10]	97
6.3.4.65 set() [9/10]	97
6.3.4.66 set() [10/10]	98
6.3.4.67 set_owner()	98
6.3.4.68 set_size() [1/10]	99
6.3.4.69 set_size() [2/10]	99
6.3.4.70 set_size() [3/10]	99
6.3.4.71 set_size() [4/10]	99
6.3.4.72 set_size() [5/10]	100
6.3.4.73 set_size() [6/10]	100
6.3.4.74 set_size() [7/10]	100
6.3.4.75 set_size() [8/10]	101
6.3.4.76 set_size() [9/10]	101

6.3.4.77 <code>set_size()</code> [10/10]	102
6.3.4.78 <code>size()</code> [1/2]	102
6.3.4.79 <code>size()</code> [2/2]	102
6.3.5 Member Data Documentation	102
6.3.5.1 <code>data_</code>	103
6.3.5.2 <code>size_</code>	103
6.4 <code>coder::array< T, 2 ></code> Class Template Reference	103
6.4.1 Detailed Description	108
6.4.2 Member Typedef Documentation	108
6.4.2.1 <code>Base</code>	108
6.4.2.2 <code>size_type</code>	108
6.4.2.3 <code>value_type</code>	108
6.4.3 Constructor & Destructor Documentation	109
6.4.3.1 <code>array()</code> [1/4]	109
6.4.3.2 <code>array()</code> [2/4]	109
6.4.3.3 <code>array()</code> [3/4]	109
6.4.3.4 <code>array()</code> [4/4]	109
6.4.4 Member Function Documentation	110
6.4.4.1 <code>at()</code> [1/20]	110
6.4.4.2 <code>at()</code> [2/20]	110
6.4.4.3 <code>at()</code> [3/20]	110
6.4.4.4 <code>at()</code> [4/20]	110
6.4.4.5 <code>at()</code> [5/20]	111
6.4.4.6 <code>at()</code> [6/20]	111
6.4.4.7 <code>at()</code> [7/20]	111
6.4.4.8 <code>at()</code> [8/20]	111
6.4.4.9 <code>at()</code> [9/20]	112
6.4.4.10 <code>at()</code> [10/20]	112
6.4.4.11 <code>at()</code> [11/20]	112
6.4.4.12 <code>at()</code> [12/20]	113
6.4.4.13 <code>at()</code> [13/20]	113
6.4.4.14 <code>at()</code> [14/20]	113
6.4.4.15 <code>at()</code> [15/20]	114
6.4.4.16 <code>at()</code> [16/20]	114
6.4.4.17 <code>at()</code> [17/20]	114
6.4.4.18 <code>at()</code> [18/20]	115
6.4.4.19 <code>at()</code> [19/20]	115
6.4.4.20 <code>at()</code> [20/20]	115
6.4.4.21 <code>begin()</code> [1/2]	116
6.4.4.22 <code>begin()</code> [2/2]	116
6.4.4.23 <code>capacity()</code>	116
6.4.4.24 <code>clear()</code>	116

6.4.4.25 <code>data()</code> [1/2]	116
6.4.4.26 <code>data()</code> [2/2]	117
6.4.4.27 <code>end()</code> [1/2]	117
6.4.4.28 <code>end()</code> [2/2]	117
6.4.4.29 <code>ensureCapacity()</code>	117
6.4.4.30 <code>index()</code> [1/10]	118
6.4.4.31 <code>index()</code> [2/10]	118
6.4.4.32 <code>index()</code> [3/10]	118
6.4.4.33 <code>index()</code> [4/10]	118
6.4.4.34 <code>index()</code> [5/10]	119
6.4.4.35 <code>index()</code> [6/10]	119
6.4.4.36 <code>index()</code> [7/10]	119
6.4.4.37 <code>index()</code> [8/10]	120
6.4.4.38 <code>index()</code> [9/10]	120
6.4.4.39 <code>index()</code> [10/10]	120
6.4.4.40 <code>is_owner()</code>	121
6.4.4.41 <code>numel()</code>	121
6.4.4.42 <code>operator std::vector< T >()</code>	121
6.4.4.43 <code>operator=()</code>	122
6.4.4.44 <code>operator[]()</code> [1/2]	122
6.4.4.45 <code>operator[]()</code> [2/2]	122
6.4.4.46 <code>reshape()</code> [1/10]	123
6.4.4.47 <code>reshape()</code> [2/10]	123
6.4.4.48 <code>reshape()</code> [3/10]	123
6.4.4.49 <code>reshape()</code> [4/10]	123
6.4.4.50 <code>reshape()</code> [5/10]	124
6.4.4.51 <code>reshape()</code> [6/10]	124
6.4.4.52 <code>reshape()</code> [7/10]	124
6.4.4.53 <code>reshape()</code> [8/10]	125
6.4.4.54 <code>reshape()</code> [9/10]	125
6.4.4.55 <code>reshape()</code> [10/10]	125
6.4.4.56 <code>reshape_n()</code>	126
6.4.4.57 <code>set()</code> [1/10]	126
6.4.4.58 <code>set()</code> [2/10]	126
6.4.4.59 <code>set()</code> [3/10]	126
6.4.4.60 <code>set()</code> [4/10]	127
6.4.4.61 <code>set()</code> [5/10]	127
6.4.4.62 <code>set()</code> [6/10]	127
6.4.4.63 <code>set()</code> [7/10]	128
6.4.4.64 <code>set()</code> [8/10]	128
6.4.4.65 <code>set()</code> [9/10]	128
6.4.4.66 <code>set()</code> [10/10]	129

6.4.4.67 <code>set_owner()</code>	129
6.4.4.68 <code>set_size()</code> [1/10]	130
6.4.4.69 <code>set_size()</code> [2/10]	130
6.4.4.70 <code>set_size()</code> [3/10]	130
6.4.4.71 <code>set_size()</code> [4/10]	130
6.4.4.72 <code>set_size()</code> [5/10]	131
6.4.4.73 <code>set_size()</code> [6/10]	131
6.4.4.74 <code>set_size()</code> [7/10]	131
6.4.4.75 <code>set_size()</code> [8/10]	132
6.4.4.76 <code>set_size()</code> [9/10]	132
6.4.4.77 <code>set_size()</code> [10/10]	133
6.4.4.78 <code>size()</code> [1/2]	133
6.4.4.79 <code>size()</code> [2/2]	133
6.4.5 Member Data Documentation	133
6.4.5.1 <code>data_</code>	134
6.4.5.2 <code>size_</code>	134
6.5 <code>coder::array_base< T, SZ, N ></code> Class Template Reference	134
6.5.1 Detailed Description	136
6.5.2 Member Typedef Documentation	137
6.5.2.1 <code>size_type</code>	137
6.5.2.2 <code>value_type</code>	137
6.5.3 Constructor & Destructor Documentation	137
6.5.3.1 <code>array_base()</code> [1/2]	137
6.5.3.2 <code>array_base()</code> [2/2]	137
6.5.4 Member Function Documentation	138
6.5.4.1 <code>at()</code> [1/20]	138
6.5.4.2 <code>at()</code> [2/20]	138
6.5.4.3 <code>at()</code> [3/20]	138
6.5.4.4 <code>at()</code> [4/20]	138
6.5.4.5 <code>at()</code> [5/20]	139
6.5.4.6 <code>at()</code> [6/20]	139
6.5.4.7 <code>at()</code> [7/20]	139
6.5.4.8 <code>at()</code> [8/20]	139
6.5.4.9 <code>at()</code> [9/20]	140
6.5.4.10 <code>at()</code> [10/20]	140
6.5.4.11 <code>at()</code> [11/20]	140
6.5.4.12 <code>at()</code> [12/20]	141
6.5.4.13 <code>at()</code> [13/20]	141
6.5.4.14 <code>at()</code> [14/20]	141
6.5.4.15 <code>at()</code> [15/20]	142
6.5.4.16 <code>at()</code> [16/20]	142
6.5.4.17 <code>at()</code> [17/20]	142

6.5.4.18 at() [18/20]	143
6.5.4.19 at() [19/20]	143
6.5.4.20 at() [20/20]	143
6.5.4.21 begin() [1/2]	144
6.5.4.22 begin() [2/2]	144
6.5.4.23 capacity()	144
6.5.4.24 clear()	144
6.5.4.25 data() [1/2]	145
6.5.4.26 data() [2/2]	145
6.5.4.27 end() [1/2]	145
6.5.4.28 end() [2/2]	145
6.5.4.29 ensureCapacity()	146
6.5.4.30 index() [1/10]	146
6.5.4.31 index() [2/10]	147
6.5.4.32 index() [3/10]	147
6.5.4.33 index() [4/10]	147
6.5.4.34 index() [5/10]	148
6.5.4.35 index() [6/10]	148
6.5.4.36 index() [7/10]	148
6.5.4.37 index() [8/10]	149
6.5.4.38 index() [9/10]	149
6.5.4.39 index() [10/10]	149
6.5.4.40 is_owner()	150
6.5.4.41 numel()	150
6.5.4.42 operator=()	151
6.5.4.43 operator[]() [1/2]	151
6.5.4.44 operator[]() [2/2]	151
6.5.4.45 reshape() [1/10]	151
6.5.4.46 reshape() [2/10]	152
6.5.4.47 reshape() [3/10]	152
6.5.4.48 reshape() [4/10]	152
6.5.4.49 reshape() [5/10]	152
6.5.4.50 reshape() [6/10]	153
6.5.4.51 reshape() [7/10]	153
6.5.4.52 reshape() [8/10]	153
6.5.4.53 reshape() [9/10]	154
6.5.4.54 reshape() [10/10]	154
6.5.4.55 reshape_n()	154
6.5.4.56 set() [1/10]	155
6.5.4.57 set() [2/10]	155
6.5.4.58 set() [3/10]	156
6.5.4.59 set() [4/10]	156

6.5.4.60 <code>set()</code> [5/10]	156
6.5.4.61 <code>set()</code> [6/10]	157
6.5.4.62 <code>set()</code> [7/10]	157
6.5.4.63 <code>set()</code> [8/10]	157
6.5.4.64 <code>set()</code> [9/10]	158
6.5.4.65 <code>set()</code> [10/10]	158
6.5.4.66 <code>set_owner()</code>	159
6.5.4.67 <code>set_size()</code> [1/10]	159
6.5.4.68 <code>set_size()</code> [2/10]	160
6.5.4.69 <code>set_size()</code> [3/10]	160
6.5.4.70 <code>set_size()</code> [4/10]	160
6.5.4.71 <code>set_size()</code> [5/10]	161
6.5.4.72 <code>set_size()</code> [6/10]	161
6.5.4.73 <code>set_size()</code> [7/10]	161
6.5.4.74 <code>set_size()</code> [8/10]	162
6.5.4.75 <code>set_size()</code> [9/10]	162
6.5.4.76 <code>set_size()</code> [10/10]	163
6.5.4.77 <code>size()</code> [1/2]	163
6.5.4.78 <code>size()</code> [2/2]	164
6.5.5 Member Data Documentation	164
6.5.5.1 <code>data_</code>	164
6.5.5.2 <code>size_</code>	164
6.6 <code>coder::array_iterator< T ></code> Class Template Reference	165
6.6.1 Detailed Description	167
6.6.2 Constructor & Destructor Documentation	167
6.6.2.1 <code>array_iterator()</code> [1/3]	167
6.6.2.2 <code>array_iterator()</code> [2/3]	167
6.6.2.3 <code>~array_iterator()</code>	168
6.6.2.4 <code>array_iterator()</code> [3/3]	168
6.6.3 Member Function Documentation	168
6.6.3.1 <code>operator"!=()</code>	168
6.6.3.2 <code>operator*()</code>	168
6.6.3.3 <code>operator+()</code>	169
6.6.3.4 <code>operator++()</code> [1/2]	169
6.6.3.5 <code>operator++()</code> [2/2]	169
6.6.3.6 <code>operator+=()</code>	169
6.6.3.7 <code>operator-()</code> [1/2]	170
6.6.3.8 <code>operator-()</code> [2/2]	170
6.6.3.9 <code>operator--()</code> [1/2]	170
6.6.3.10 <code>operator--()</code> [2/2]	170
6.6.3.11 <code>operator-=()</code>	171
6.6.3.12 <code>operator->()</code>	171

6.6.3.13 operator<()	171
6.6.3.14 operator<=()	171
6.6.3.15 operator=()	172
6.6.3.16 operator==()	172
6.6.3.17 operator>()	172
6.6.3.18 operator>=()	172
6.6.3.19 operator[]()	173
6.6.4 Member Data Documentation	173
6.6.4.1 arr_	173
6.6.4.2 i_	173
6.7 cell_wrap_0 Struct Reference	173
6.7.1 Detailed Description	174
6.7.2 Member Data Documentation	174
6.7.2.1 f1 [1/2]	174
6.7.2.2 f1 [2/2]	174
6.8 cell_wrap_1 Struct Reference	175
6.8.1 Detailed Description	176
6.8.2 Member Data Documentation	176
6.8.2.1 f1	176
6.9 coder::const_array_iterator< T > Class Template Reference	176
6.9.1 Detailed Description	179
6.9.2 Constructor & Destructor Documentation	179
6.9.2.1 const_array_iterator() [1/3]	179
6.9.2.2 const_array_iterator() [2/3]	179
6.9.2.3 ~const_array_iterator()	180
6.9.2.4 const_array_iterator() [3/3]	180
6.9.3 Member Function Documentation	180
6.9.3.1 operator"!="()	180
6.9.3.2 operator*()	180
6.9.3.3 operator+()	181
6.9.3.4 operator++() [1/2]	181
6.9.3.5 operator++() [2/2]	181
6.9.3.6 operator+=()	181
6.9.3.7 operator-() [1/2]	182
6.9.3.8 operator-() [2/2]	182
6.9.3.9 operator--() [1/2]	182
6.9.3.10 operator--() [2/2]	182
6.9.3.11 operator-=()	183
6.9.3.12 operator->()	183
6.9.3.13 operator<()	183
6.9.3.14 operator<=()	183
6.9.3.15 operator=()	184

6.9.3.16 operator==()	184
6.9.3.17 operator>()	184
6.9.3.18 operator>=()	184
6.9.3.19 operator[]()	185
6.9.4 Member Data Documentation	185
6.9.4.1 arr_	185
6.9.4.2 i_	185
6.9.4.3 n_	185
6.10 coder::detail::data_ptr< T, SZ > Class Template Reference	186
6.10.1 Detailed Description	187
6.10.2 Member Typedef Documentation	187
6.10.2.1 size_type	187
6.10.2.2 value_type	187
6.10.3 Constructor & Destructor Documentation	188
6.10.3.1 data_ptr() [1/3]	188
6.10.3.2 data_ptr() [2/3]	188
6.10.3.3 data_ptr() [3/3]	188
6.10.3.4 ~data_ptr()	189
6.10.4 Member Function Documentation	189
6.10.4.1 capacity()	189
6.10.4.2 clear()	189
6.10.4.3 copy() [1/2]	189
6.10.4.4 copy() [2/2]	190
6.10.4.5 is_null()	190
6.10.4.6 is_owner()	191
6.10.4.7 operator const T *()	191
6.10.4.8 operator T*()	191
6.10.4.9 operator->() [1/2]	191
6.10.4.10 operator->() [2/2]	191
6.10.4.11 operator=()	192
6.10.4.12 operator[]() [1/2]	192
6.10.4.13 operator[]() [2/2]	192
6.10.4.14 reserve()	192
6.10.4.15 resize()	193
6.10.4.16 set()	193
6.10.4.17 set_owner()	194
6.10.5 Member Data Documentation	194
6.10.5.1 capacity_	194
6.10.5.2 data_	194
6.10.5.3 owner_	194
6.10.5.4 size_	194
6.11 coder::detail::index_nd< I > Class Template Reference	195

6.11.1 Detailed Description	195
6.11.2 Member Function Documentation	195
6.11.2.1 compute()	195
6.12 coder::detail::index_nd< 0 > Class Reference	196
6.12.1 Detailed Description	196
6.12.2 Member Function Documentation	196
6.12.2.1 compute()	197
6.13 coder::detail::match_dimensions< Cond > Struct Template Reference	197
6.13.1 Detailed Description	197
6.14 coder::detail::match_dimensions< true > Struct Reference	198
6.14.1 Detailed Description	198
6.14.2 Member Function Documentation	198
6.14.2.1 check()	198
6.15 coder::detail::numel< N > Class Template Reference	199
6.15.1 Detailed Description	199
6.15.2 Member Function Documentation	199
6.15.2.1 compute()	199
6.16 coder::detail::numel< 0 > Class Reference	200
6.16.1 Detailed Description	200
6.16.2 Member Function Documentation	200
6.16.2.1 compute()	201
6.17 struct_T Struct Reference	201
6.17.1 Detailed Description	202
6.17.2 Member Data Documentation	202
6.17.2.1 cs	202
6.17.2.2 dc	202
6.17.2.3 dFOV	202
6.17.2.4 dvec	202
6.17.2.5 fov	202
6.17.2.6 H	203
6.17.2.7 mindis	203
6.17.2.8 radius	203
6.17.2.9 Rmin	203
6.17.2.10 scolor	203
6.17.2.11 size	203
6.17.2.12 W	203
7 File Documentation	205
7.1 codegen/lib(Func_fovPathPlanning/any1.cpp File Reference	205
7.1.1 Function Documentation	205
7.1.1.1 any()	206
7.2 codegen/lib(Func_fovPathPlanning/any1.h File Reference	206

7.2.1 Macro Definition Documentation	207
7.2.1.1 MAX_THREADS	208
7.2.2 Function Documentation	208
7.2.2.1 any()	208
7.3 codegen/lib(Func_fovPathPlanning/bsxfun.cpp File Reference	209
7.3.1 Function Documentation	209
7.3.1.1 b_bsxfun()	209
7.3.1.2 bsxfun()	210
7.3.1.3 c_bsxfun()	210
7.4 codegen/lib(Func_fovPathPlanning/bsxfun.h File Reference	211
7.4.1 Macro Definition Documentation	212
7.4.1.1 MAX_THREADS	212
7.4.2 Function Documentation	212
7.4.2.1 b_bsxfun()	213
7.4.2.2 bsxfun()	213
7.4.2.3 c_bsxfun()	214
7.5 codegen/lib(Func_fovPathPlanning/bwlabel.cpp File Reference	214
7.5.1 Function Documentation	215
7.5.1.1 b_bwlabel()	215
7.5.1.2 bwlabel()	217
7.5.1.3 c_bwlabel()	220
7.6 codegen/lib(Func_fovPathPlanning/bwlabel.h File Reference	222
7.6.1 Macro Definition Documentation	223
7.6.1.1 MAX_THREADS	224
7.6.2 Function Documentation	224
7.6.2.1 b_bwlabel()	224
7.6.2.2 bwlabel()	226
7.6.2.3 c_bwlabel()	229
7.7 codegen/lib(Func_fovPathPlanning/coder_array.h File Reference	231
7.7.1 Macro Definition Documentation	233
7.7.1.1 _mw_coder_array_h	233
7.7.1.2 CODER_ARRAY_NEW_DELETE	233
7.7.1.3 CODER_DELETE	233
7.7.1.4 CODER_NEW	233
7.8 codegen/lib(Func_fovPathPlanning/examples/main.cpp File Reference	234
7.8.1 Function Documentation	234
7.8.1.1 main()	234
7.9 codegen/lib(Func_fovPathPlanning/examples/main.h File Reference	235
7.9.1 Macro Definition Documentation	236
7.9.1.1 MAX_THREADS	236
7.9.2 Function Documentation	236
7.9.2.1 main()	236

7.10 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning.cpp File Reference	236
7.10.1 Function Documentation	237
7.10.1.1 Func_fovPathPlanning()	237
7.11 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning.h File Reference	240
7.11.1 Macro Definition Documentation	241
7.11.1.1 MAX_THREADS	241
7.11.2 Function Documentation	242
7.11.2.1 Func_fovPathPlanning()	242
7.12 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_data.cpp File Reference	245
7.12.1 Variable Documentation	246
7.12.1.1 bv	246
7.12.1.2 bv1	246
7.12.1.3 bv2	247
7.12.1.4 bv3	247
7.12.1.5 emlrltNestLockGlobal	247
7.12.1.6 isInitialized_Func_fovPathPlanning	247
7.13 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_data.h File Reference	248
7.13.1 Macro Definition Documentation	249
7.13.1.1 MAX_THREADS	249
7.13.2 Variable Documentation	249
7.13.2.1 bv	249
7.13.2.2 bv1	249
7.13.2.3 bv2	249
7.13.2.4 bv3	249
7.13.2.5 emlrltNestLockGlobal	250
7.13.2.6 isInitialized_Func_fovPathPlanning	250
7.14 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_initialize.cpp File Reference	250
7.14.1 Function Documentation	250
7.14.1.1 Func_fovPathPlanning_initialize()	251
7.15 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_initialize.h File Reference	251
7.15.1 Macro Definition Documentation	252
7.15.1.1 MAX_THREADS	252
7.15.2 Function Documentation	252
7.15.2.1 Func_fovPathPlanning_initialize()	252
7.16 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_rtwutil.cpp File Reference	253
7.16.1 Function Documentation	253
7.16.1.1 rt_atan2d_snf()	253
7.16.1.2 rt_remd_snf()	254
7.16.1.3 rt_roundd_snf()	256
7.17 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_rtwutil.h File Reference	256
7.17.1 Macro Definition Documentation	257
7.17.1.1 MAX_THREADS	257

7.17.2 Function Documentation	258
7.17.2.1 rt_atan2d_snf()	258
7.17.2.2 rt_remd_snf()	259
7.17.2.3 rt_roundd_snf()	260
7.18 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_terminate.cpp File Reference	261
7.18.1 Function Documentation	261
7.18.1.1 Func_fovPathPlanning_terminate()	262
7.19 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_terminate.h File Reference	262
7.19.1 Macro Definition Documentation	263
7.19.1.1 MAX_THREADS	263
7.19.2 Function Documentation	263
7.19.2.1 Func_fovPathPlanning_terminate()	263
7.20 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_types.h File Reference	264
7.20.1 Macro Definition Documentation	264
7.20.1.1 MAX_THREADS	264
7.21 codegen/lib(Func_fovPathPlanning/imclose.cpp File Reference	265
7.21.1 Function Documentation	265
7.21.1.1 b_imclose()	265
7.21.1.2 imclose()	266
7.22 codegen/lib(Func_fovPathPlanning/imclose.h File Reference	267
7.22.1 Macro Definition Documentation	268
7.22.1.1 MAX_THREADS	268
7.22.2 Function Documentation	269
7.22.2.1 b_imclose()	269
7.22.2.2 imclose()	270
7.23 codegen/lib(Func_fovPathPlanning/imdilate.cpp File Reference	271
7.23.1 Function Documentation	271
7.23.1.1 b_imdilate()	272
7.23.1.2 imdilate()	273
7.24 codegen/lib(Func_fovPathPlanning/imdilate.h File Reference	274
7.24.1 Macro Definition Documentation	275
7.24.1.1 MAX_THREADS	276
7.24.2 Function Documentation	276
7.24.2.1 b_imdilate()	276
7.24.2.2 imdilate()	277
7.25 codegen/lib(Func_fovPathPlanning/imerode.cpp File Reference	278
7.25.1 Function Documentation	279
7.25.1.1 b_imerode()	279
7.25.1.2 c_imerode()	281
7.25.1.3 imerode()	282
7.26 codegen/lib(Func_fovPathPlanning/imerode.h File Reference	284
7.26.1 Macro Definition Documentation	285

7.26.1.1 MAX_THREADS	285
7.26.2 Function Documentation	285
7.26.2.1 b_imerode()	285
7.26.2.2 c_imerode()	287
7.26.2.3 imerode()	288
7.27 codegen/lib(Func_fovPathPlanning/imopen.cpp File Reference	290
7.27.1 Function Documentation	290
7.27.1.1 imopen()	290
7.28 codegen/lib(Func_fovPathPlanning/imopen.h File Reference	292
7.28.1 Macro Definition Documentation	293
7.28.1.1 MAX_THREADS	293
7.28.2 Function Documentation	293
7.28.2.1 imopen()	293
7.29 codegen/lib(Func_fovPathPlanning/imresize.cpp File Reference	295
7.29.1 Function Documentation	295
7.29.1.1 b_imresize()	296
7.29.1.2 imresize()	296
7.30 codegen/lib(Func_fovPathPlanning/imresize.h File Reference	297
7.30.1 Macro Definition Documentation	298
7.30.1.1 MAX_THREADS	299
7.30.2 Function Documentation	299
7.30.2.1 b_imresize()	299
7.30.2.2 imresize()	300
7.31 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.c File Reference	301
7.31.1 Function Documentation	301
7.31.1.1 Func_fovPathPlanning_api()	302
7.31.1.2 Func_fovPathPlanning_atexit()	302
7.31.1.3 Func_fovPathPlanning_initialize()	303
7.31.1.4 Func_fovPathPlanning_terminate()	304
7.31.2 Variable Documentation	304
7.31.2.1 emlrtContextGlobal	304
7.31.2.2 emlrtRootTLSGlobal	305
7.32 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.h File Reference	305
7.32.1 Macro Definition Documentation	306
7.32.1.1 MAX_THREADS	306
7.32.1.2 typedef_cell_wrap_0	306
7.32.2 Function Documentation	306
7.32.2.1 Func_fovPathPlanning()	307
7.32.2.2 Func_fovPathPlanning_api()	307
7.32.2.3 Func_fovPathPlanning_atexit()	308
7.32.2.4 Func_fovPathPlanning_initialize()	309
7.32.2.5 Func_fovPathPlanning_terminate()	309

7.32.2.6 Func_fovPathPlanning_xil_shutdown()	310
7.32.2.7 Func_fovPathPlanning_xil_terminate()	310
7.32.3 Variable Documentation	310
7.32.3.1 emlrtContextGlobal	311
7.32.3.2 emlrtRootTLSGlobal	311
7.33 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_info.c File Reference	311
7.33.1 Function Documentation	311
7.33.1.1 emlrtMexFcnProperties()	312
7.34 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_info.h File Reference	312
7.34.1 Macro Definition Documentation	313
7.34.1.1 MAX_THREADS	313
7.34.2 Function Documentation	313
7.34.2.1 emlrtMexFcnProperties()	314
7.35 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_mex.c File Reference	314
7.35.1 Function Documentation	315
7.35.1.1 c_Func_fovPathPlanning_mexFunct()	315
7.35.1.2 mexFunction()	316
7.35.1.3 mexFunctionCreateRootTLS()	317
7.36 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_mex.h File Reference	317
7.36.1 Macro Definition Documentation	318
7.36.1.1 MAX_THREADS	318
7.36.2 Function Documentation	318
7.36.2.1 mexFunction()	319
7.36.2.2 mexFunctionCreateRootTLS()	319
7.37 codegen/lib(Func_fovPathPlanning/kw_CBGA.cpp File Reference	320
7.37.1 Function Documentation	320
7.37.1.1 kw_CBGA()	321
7.38 codegen/lib(Func_fovPathPlanning/kw_CBGA.h File Reference	327
7.38.1 Macro Definition Documentation	328
7.38.1.1 MAX_THREADS	328
7.38.2 Function Documentation	328
7.38.2.1 kw_CBGA()	328
7.39 codegen/lib(Func_fovPathPlanning/kw_cellDecomposition.cpp File Reference	334
7.39.1 Function Documentation	335
7.39.1.1 b_kw_cellDecomposition()	335
7.39.1.2 kw_cellDecomposition()	337
7.40 codegen/lib(Func_fovPathPlanning/kw_cellDecomposition.h File Reference	340
7.40.1 Macro Definition Documentation	341
7.40.1.1 MAX_THREADS	342
7.40.2 Function Documentation	342
7.40.2.1 b_kw_cellDecomposition()	342
7.40.2.2 kw_cellDecomposition()	344

7.41 codegen/lib(Func_fovPathPlanning/kw_point2ind.cpp File Reference	347
7.41.1 Function Documentation	348
7.41.1.1 kw_point2ind()	348
7.42 codegen/lib(Func_fovPathPlanning/kw_point2ind.h File Reference	350
7.42.1 Macro Definition Documentation	351
7.42.1.1 MAX_THREADS	351
7.42.2 Function Documentation	351
7.42.2.1 kw_point2ind()	352
7.43 codegen/lib(Func_fovPathPlanning/KWcalcMapCoverage.cpp File Reference	354
7.43.1 Function Documentation	354
7.43.1.1 b_KWcalcMapCoverage()	354
7.43.1.2 KWcalcMapCoverage()	357
7.44 codegen/lib(Func_fovPathPlanning/KWcalcMapCoverage.h File Reference	359
7.44.1 Macro Definition Documentation	360
7.44.1.1 MAX_THREADS	361
7.44.2 Function Documentation	361
7.44.2.1 b_KWcalcMapCoverage()	361
7.44.2.2 KWcalcMapCoverage()	363
7.45 codegen/lib(Func_fovPathPlanning/KWcontactCell.cpp File Reference	366
7.45.1 Function Documentation	367
7.45.1.1 KWcontactCell()	367
7.46 codegen/lib(Func_fovPathPlanning/KWcontactCell.h File Reference	369
7.46.1 Macro Definition Documentation	370
7.46.1.1 MAX_THREADS	371
7.46.2 Function Documentation	371
7.46.2.1 KWcontactCell()	371
7.47 codegen/lib(Func_fovPathPlanning/KWMapCoverage.cpp File Reference	373
7.47.1 Function Documentation	374
7.47.1.1 KWMapCoverage()	374
7.48 codegen/lib(Func_fovPathPlanning/KWMapCoverage.h File Reference	376
7.48.1 Macro Definition Documentation	377
7.48.1.1 MAX_THREADS	377
7.48.2 Function Documentation	377
7.48.2.1 KWMapCoverage()	377
7.49 codegen/lib(Func_fovPathPlanning/linspace.cpp File Reference	379
7.49.1 Function Documentation	379
7.49.1.1 linspace()	379
7.50 codegen/lib(Func_fovPathPlanning/linspace.h File Reference	380
7.50.1 Macro Definition Documentation	381
7.50.1.1 MAX_THREADS	382
7.50.2 Function Documentation	382
7.50.2.1 linspace()	382

7.51 codegen/lib(Func_fovPathPlanning/rtDefines.h File Reference	383
7.52 codegen/lib(Func_fovPathPlanning/rt_nonfinite.cpp File Reference	384
7.52.1 Function Documentation	385
7.52.1.1 rt_InitInfAndNaN()	385
7.52.1.2 rtIsInf()	386
7.52.1.3 rtIsInfF()	386
7.52.1.4 rtIsNaN()	386
7.52.1.5 rtIsNaNF()	387
7.52.2 Variable Documentation	387
7.52.2.1 rtInf	387
7.52.2.2 rtInfF	387
7.52.2.3 rtMinusInf	388
7.52.2.4 rtMinusInfF	388
7.52.2.5 rtNaN	388
7.52.2.6 rtNaNF	388
7.53 codegen/lib(Func_fovPathPlanning/rt_nonfinite.h File Reference	388
7.53.1 Function Documentation	389
7.53.1.1 rt_InitInfAndNaN()	389
7.53.1.2 rtIsInf()	390
7.53.1.3 rtIsInfF()	390
7.53.1.4 rtIsNaN()	390
7.53.1.5 rtIsNaNF()	391
7.53.2 Variable Documentation	391
7.53.2.1 rtInf	391
7.53.2.2 rtInfF	391
7.53.2.3 rtMinusInf	392
7.53.2.4 rtMinusInfF	392
7.53.2.5 rtNaN	392
7.53.2.6 rtNaNF	392
7.54 codegen/lib(Func_fovPathPlanning/rtGetInf.cpp File Reference	393
7.54.1 Function Documentation	393
7.54.1.1 rtGetInf()	393
7.54.1.2 rtGetInfF()	394
7.54.1.3 rtGetMinusInf()	394
7.54.1.4 rtGetMinusInfF()	394
7.55 codegen/lib(Func_fovPathPlanning/rtGetInf.h File Reference	394
7.55.1 Function Documentation	395
7.55.1.1 rtGetInf()	396
7.55.1.2 rtGetInfF()	396
7.55.1.3 rtGetMinusInf()	396
7.55.1.4 rtGetMinusInfF()	396
7.56 codegen/lib(Func_fovPathPlanning/rtGetNaN.cpp File Reference	397

7.56.1 Function Documentation	397
7.56.1.1 rtGetNaN()	397
7.56.1.2 rtGetNaNF()	398
7.57 codegen/lib(Func_fovPathPlanning/rtGetNaN.h File Reference	398
7.57.1 Function Documentation	399
7.57.1.1 rtGetNaN()	399
7.57.1.2 rtGetNaNF()	399
7.58 codegen/lib(Func_fovPathPlanning/rtwtypes.h File Reference	400

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<code>coder</code>	9
<code>coder::detail</code>	10

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

coder::array_base< T, SZ, N >	134
coder::array_base< char_T, SizeType, 2 >	134
coder::array< char_T, 2 >	40
coder::array_base< double, SizeType, N >	134
coder::array< double, 2U >	11
coder::array_base< T, SizeType, 1 >	134
coder::array< T, 1 >	73
coder::array_base< T, SizeType, 2 >	134
coder::array< T, 2 >	103
coder::array_base< T, SizeType, N >	134
coder::array< T, N >	11
cell_wrap_0	173
cell_wrap_1	175
coder::detail::data_ptr< T, SZ >	186
coder::detail::data_ptr< char_T, SizeType >	186
coder::detail::data_ptr< double, SizeType >	186
coder::detail::data_ptr< T, SizeType >	186
coder::detail::index_nd< I >	195
coder::detail::index_nd< 0 >	196
iterator	
coder::array_iterator< T >	165
coder::const_array_iterator< T >	176
coder::detail::match_dimensions< Cond >	197
coder::detail::match_dimensions< true >	198
coder::detail::numel< N >	199
coder::detail::numel< 0 >	200
struct_T	201

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

coder::array< T, N >	11
coder::array< char_T, 2 >	40
coder::array< T, 1 >	73
coder::array< T, 2 >	103
coder::array_base< T, SZ, N >	134
coder::array_iterator< T >	165
cell_wrap_0	173
cell_wrap_1	175
coder::const_array_iterator< T >	176
coder::detail::data_ptr< T, SZ >	186
coder::detail::index_nd< I >	195
coder::detail::index_nd< 0 >	196
coder::detail::match_dimensions< Cond >	197
coder::detail::match_dimensions< true >	198
coder::detail::numel< N >	199
coder::detail::numel< 0 >	200
struct_T	201

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

codegen/lib/Func_fovPathPlanning/ any1.cpp	205
codegen/lib/Func_fovPathPlanning/ any1.h	206
codegen/lib/Func_fovPathPlanning/ bsxfun.cpp	209
codegen/lib/Func_fovPathPlanning/ bsxfun.h	211
codegen/lib/Func_fovPathPlanning/ bwlabel.cpp	214
codegen/lib/Func_fovPathPlanning/ bwlabel.h	222
codegen/lib/Func_fovPathPlanning/ coder_array.h	231
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning.cpp	236
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning.h	240
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_data.cpp	245
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_data.h	248
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_initialize.cpp	250
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_initialize.h	251
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_rtwutil.cpp	253
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_rtwutil.h	256
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_terminate.cpp	261
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_terminate.h	262
codegen/lib/Func_fovPathPlanning/ Func_fovPathPlanning_types.h	264
codegen/lib/Func_fovPathPlanning/ imclose.cpp	265
codegen/lib/Func_fovPathPlanning/ imclose.h	267
codegen/lib/Func_fovPathPlanning/ imdilate.cpp	271
codegen/lib/Func_fovPathPlanning/ imdilate.h	274
codegen/lib/Func_fovPathPlanning/ imerode.cpp	278
codegen/lib/Func_fovPathPlanning/ imerode.h	284
codegen/lib/Func_fovPathPlanning/ imopen.cpp	290
codegen/lib/Func_fovPathPlanning/ imopen.h	292
codegen/lib/Func_fovPathPlanning/ imresize.cpp	295
codegen/lib/Func_fovPathPlanning/ imresize.h	297
codegen/lib/Func_fovPathPlanning/ kw_CBGA.cpp	320
codegen/lib/Func_fovPathPlanning/ kw_CBGA.h	327
codegen/lib/Func_fovPathPlanning/ kw_cellDecomposition.cpp	334
codegen/lib/Func_fovPathPlanning/ kw_cellDecomposition.h	340
codegen/lib/Func_fovPathPlanning/ kw_point2ind.cpp	347
codegen/lib/Func_fovPathPlanning/ kw_point2ind.h	350
codegen/lib/Func_fovPathPlanning/ KWcalcMapCoverage.cpp	354

codegen/lib/Func_fovPathPlanning/ KWcalcMapCoverage.h	359
codegen/lib/Func_fovPathPlanning/ KWcontactCell.cpp	366
codegen/lib/Func_fovPathPlanning/ KWcontactCell.h	369
codegen/lib/Func_fovPathPlanning/ KWMapCoverage.cpp	373
codegen/lib/Func_fovPathPlanning/ KWMapCoverage.h	376
codegen/lib/Func_fovPathPlanning/ linspace.cpp	379
codegen/lib/Func_fovPathPlanning/ linspace.h	380
codegen/lib/Func_fovPathPlanning/ rtDefines.h	383
codegen/lib/Func_fovPathPlanning/ rtNonfinite.cpp	384
codegen/lib/Func_fovPathPlanning/ rtNonfinite.h	388
codegen/lib/Func_fovPathPlanning/ rtGetInf.cpp	393
codegen/lib/Func_fovPathPlanning/ rtGetInf.h	394
codegen/lib/Func_fovPathPlanning/ rtGetNaN.cpp	397
codegen/lib/Func_fovPathPlanning/ rtGetNaN.h	398
codegen/lib/Func_fovPathPlanning/ rtwtypes.h	400
codegen/lib/Func_fovPathPlanning/examples/ main.cpp	234
codegen/lib/Func_fovPathPlanning/examples/ main.h	235
codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.c	301
codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.h	305
codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_info.c	311
codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_info.h	312
codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_mex.c	314
codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_mex.h	317

Chapter 5

Namespace Documentation

5.1 coder Namespace Reference

Namespaces

- [detail](#)

Classes

- class [array](#)
- class [array< char_T, 2 >](#)
- class [array< T, 1 >](#)
- class [array< T, 2 >](#)
- class [array_base](#)
- class [array_iterator](#)
- class [const_array_iterator](#)

TypeDefs

- [typedef int32_T SizeType](#)

5.1.1 TypeDef Documentation

5.1.1.1 SizeType

```
typedef int32_T coder::SizeType
```

Definition at line 60 of file `coder_array.h`.

5.2 coder::detail Namespace Reference

Classes

- class [data_ptr](#)
- class [index_nd](#)
- class [index_nd< 0 >](#)
- struct [match_dimensions](#)
- struct [match_dimensions< true >](#)
- class [numel](#)
- class [numel< 0 >](#)

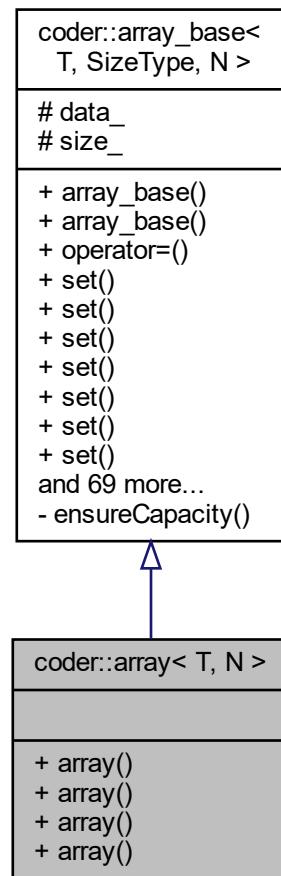
Chapter 6

Class Documentation

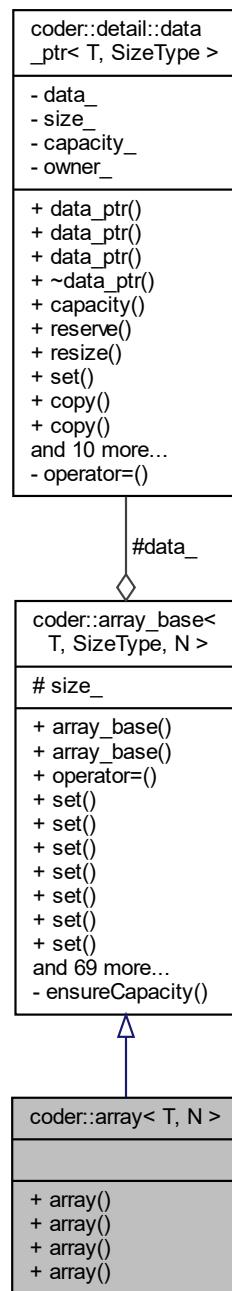
6.1 coder::array< T, N > Class Template Reference

```
#include <coder_array.h>
```

Inheritance diagram for coder::array< T, N >:



Collaboration diagram for coder::array< T, N >:



Public Types

- `typedef T value_type`
- `typedef SizeType size_type`

Public Member Functions

- `array ()`

- `array (const array< T, N > &_other)`
- `array (const Base &_other)`
- `array (T *_data, const SizeType *_sz)`
- `void set (T *_data, SizeType _n1)`
- `void set (T *_data, SizeType _n1, SizeType _n2)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `bool is_owner () const`
- `void set_owner (bool b)`
- `SizeType capacity () const`
- `void set_size (SizeType _n1)`
- `void set_size (SizeType _n1, SizeType _n2)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `array_base< T, SizeType, N1 > reshape_n (const SizeType(&_ns)[N1]) const`
- `array_base< T, SizeType, 1 > reshape (SizeType _n1) const`
- `array_base< T, SizeType, 2 > reshape (SizeType _n1, SizeType _n2) const`
- `array_base< T, SizeType, 3 > reshape (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `array_base< T, SizeType, 4 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `array_base< T, SizeType, 5 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `array_base< T, SizeType, 6 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `array_base< T, SizeType, 7 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `array_base< T, SizeType, 8 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `array_base< T, SizeType, 9 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`
- `array_base< T, SizeType, 10 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- `T & operator[] (SizeType _index)`
- `const T & operator[] (SizeType _index) const`
- `void clear ()`
- `T * data ()`

- const T * `data()` const
- const `SizeType * size()` const
- `SizeType size(SizeType _index)` const
- `SizeType numel()` const
- `SizeType index(SizeType _n1)` const
- `SizeType index(SizeType _n1, SizeType _n2)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)` const
- `SizeType index(SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)` const
- `T & at(SizeType _i1)`
- `T & at(SizeType _i1, SizeType _i2)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9)`
- `T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10)`
- `const T & at(SizeType _i1)` const
- `const T & at(SizeType _i1, SizeType _i2)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9)` const
- `const T & at(SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10)` const
- `array_iterator< array_base< T, SizeType, N > > begin()`
- `const_array_iterator< array_base< T, SizeType, N > > begin()` const
- `array_iterator< array_base< T, SizeType, N > > end()`
- `const_array_iterator< array_base< T, SizeType, N > > end()` const

Protected Attributes

- `::coder::detail::data_ptr< T, SizeType > data_`
- `SizeType size_[N]`

Private Types

- `typedef array_base< T, SizeType, N > Base`

Private Member Functions

- `void ensureCapacity (SizeType _newNumel)`

6.1.1 Detailed Description

```
template<typename T, int N>
class coder::array< T, N >
```

Definition at line 970 of file coder_array.h.

6.1.2 Member Typedef Documentation

6.1.2.1 Base

```
template<typename T , int N>
typedef array_base<T, SizeType, N> coder::array< T, N >::Base [private]
```

Definition at line 972 of file coder_array.h.

6.1.2.2 size_type

```
typedef SizeType coder::array_base< T, SizeType , N >::size_type [inherited]
```

Definition at line 464 of file coder_array.h.

6.1.2.3 value_type

```
typedef T coder::array_base< T, SizeType , N >::value_type [inherited]
```

Definition at line 463 of file coder_array.h.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 array() [1/4]

```
template<typename T , int N>
coder::array< T, N >::array ( )  [inline]
```

Definition at line 975 of file coder_array.h.

```
976     : Base() {
977 }
```

6.1.3.2 array() [2/4]

```
template<typename T , int N>
coder::array< T, N >::array (
    const array< T, N > & _other )  [inline]
```

Definition at line 978 of file coder_array.h.

```
979     : Base(_other) {
980 }
```

6.1.3.3 array() [3/4]

```
template<typename T , int N>
coder::array< T, N >::array (
    const Base & _other )  [inline]
```

Definition at line 981 of file coder_array.h.

```
982     : Base(_other) {
983 }
```

6.1.3.4 array() [4/4]

```
template<typename T , int N>
coder::array< T, N >::array (
    T * _data,
    const SizeType * _sz )  [inline]
```

Definition at line 984 of file coder_array.h.

```
985     : Base(_data, _sz) {
986 }
```

6.1.4 Member Function Documentation

6.1.4.1 at() [1/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1 ) [inline], [inherited]
```

Definition at line 847 of file coder_array.h.

```
847     {
848         ::coder::detail::match_dimensions<N == 1>::check();
849         return data_[_i1];
850     }
```

6.1.4.2 at() [2/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1 ) const [inline], [inherited]
```

Definition at line 888 of file coder_array.h.

```
888     {
889         ::coder::detail::match_dimensions<N == 1>::check();
890         return data_[_i1];
891     }
```

6.1.4.3 at() [3/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) [inline], [inherited]
```

Definition at line 851 of file coder_array.h.

```
851     {
852         ::coder::detail::match_dimensions<N == 2>::check();
853         return data_[index(_i1, _i2)];
854     }
```

6.1.4.4 at() [4/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) const [inline], [inherited]
```

Definition at line 892 of file coder_array.h.

```
892     {
893         ::coder::detail::match_dimensions<N == 2>::check();
894         return data_[index(_i1, _i2)];
895     }
```

6.1.4.5 at() [5/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) [inline], [inherited]
```

Definition at line 855 of file coder_array.h.

```
855             {
856         ::coder::detail::match_dimensions<N == 3>::check();
857         return data_[index(_i1, _i2, _i3)];
858     }
```

6.1.4.6 at() [6/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) const [inline], [inherited]
```

Definition at line 896 of file coder_array.h.

```
896             {
897         ::coder::detail::match_dimensions<N == 3>::check();
898         return data_[index(_i1, _i2, _i3)];
899     }
```

6.1.4.7 at() [7/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) [inline], [inherited]
```

Definition at line 859 of file coder_array.h.

```
859             {
860         ::coder::detail::match_dimensions<N == 4>::check();
861         return data_[index(_i1, _i2, _i3, _i4)];
862     }
```

6.1.4.8 at() [8/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) const [inline], [inherited]
```

Definition at line 900 of file coder_array.h.

```
900             {
901         ::coder::detail::match_dimensions<N == 4>::check();
902         return data_[index(_i1, _i2, _i3, _i4)];
903     }
```

6.1.4.9 at() [9/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) [inline], [inherited]
```

Definition at line 863 of file coder_array.h.

```
863     ::coder::detail::match_dimensions<N == 5>::check();
864     return data_[index(_i1, _i2, _i3, _i4, _i5)];
```

6.1.4.10 at() [10/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) const [inline], [inherited]
```

Definition at line 904 of file coder_array.h.

```
904     ::coder::detail::match_dimensions<N == 5>::check();
905     return data_[index(_i1, _i2, _i3, _i4, _i5)];
```

6.1.4.11 at() [11/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) [inline], [inherited]
```

Definition at line 867 of file coder_array.h.

```
867     ::coder::detail::match_dimensions<N == 6>::check();
868     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
```

6.1.4.12 at() [12/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) const [inline], [inherited]
```

Definition at line 908 of file coder_array.h.

```
908     ::coder::detail::match_dimensions<N == 6>::check();
909     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
910 }
911 }
```

6.1.4.13 at() [13/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) [inline], [inherited]
```

Definition at line 871 of file coder_array.h.

```
871     ::coder::detail::match_dimensions<N == 7>::check();
872     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
873 }
874 }
```

6.1.4.14 at() [14/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) const [inline], [inherited]
```

Definition at line 912 of file coder_array.h.

```
912     ::coder::detail::match_dimensions<N == 7>::check();
913     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
914 }
915 }
```

6.1.4.15 at() [15/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) [inline], [inherited]
```

Definition at line 875 of file coder_array.h.

```
875
876     ::coder::detail::match_dimensions<N == 8>::check();
877     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
878 }
```

6.1.4.16 at() [16/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) const [inline], [inherited]
```

Definition at line 916 of file coder_array.h.

```
916
917     ::coder::detail::match_dimensions<N == 8>::check();
918     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
919 }
```

6.1.4.17 at() [17/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) [inline], [inherited]
```

Definition at line 879 of file coder_array.h.

```
879
880     ::coder::detail::match_dimensions<N == 9>::check();
881     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
882 }
```

6.1.4.18 at() [18/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) const [inline], [inherited]
```

Definition at line 920 of file coder_array.h.

```
920
921     ::coder::detail::match_dimensions<N == 9>::check();
922     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
923 }
```

{

6.1.4.19 at() [19/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) [inline], [inherited]
```

Definition at line 883 of file coder_array.h.

```
883
884     ::coder::detail::match_dimensions<N == 10>::check();
885     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
886 }
```

{

6.1.4.20 at() [20/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) const [inline], [inherited]
```

Definition at line 924 of file coder_array.h.

```
925     {
926     ::coder::detail::match_dimensions<N == 10>::check();
927     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
928 }
```

6.1.4.21 begin() [1/2]

```
array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::begin  
[inline], [inherited]
```

Definition at line 930 of file coder_array.h.

```
930 {  
931     return array_iterator<array_base<T, SZ, N> >(this, 0);  
932 }
```

6.1.4.22 begin() [2/2]

```
const_array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::begin  
[inline], [inherited]
```

Definition at line 936 of file coder_array.h.

```
936 {  
937     return const_array_iterator<array_base<T, SZ, N> >(this, 0);  
938 }
```

6.1.4.23 capacity()

```
SizeType coder::array_base< T, SizeType , N >::capacity [inline], [inherited]
```

Definition at line 595 of file coder_array.h.

```
595 {  
596     return data_.capacity();  
597 }
```

6.1.4.24 clear()

```
void coder::array_base< T, SizeType , N >::clear [inline], [inherited]
```

Definition at line 771 of file coder_array.h.

```
771 {  
772     data_.clear();  
773 }
```

6.1.4.25 data() [1/2]

```
T* coder::array_base< T, SizeType , N >::data [inline], [inherited]
```

Definition at line 775 of file coder_array.h.

```
775 {  
776     return data_;  
777 }
```

6.1.4.26 data() [2/2]

```
const T* coder::array_base< T, SizeType , N >::data [inline], [inherited]
```

Definition at line 779 of file coder_array.h.

```
779         {
780             return data_;
781         }
```

6.1.4.27 end() [1/2]

```
array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::end [inline], [inherited]
```

Definition at line 933 of file coder_array.h.

```
933         {
934             return array_iterator<array_base<T, SZ, N> >(this, this->numel());
935         }
```

6.1.4.28 end() [2/2]

```
const_array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::end [inline], [inherited]
```

Definition at line 939 of file coder_array.h.

```
939         {
940             return const_array_iterator<array_base<T, SZ, N> >(this, this->numel());
941         }
```

6.1.4.29 ensureCapacity()

```
void coder::array_base< T, SizeType , N >::ensureCapacity (
    SizeType _newNumel ) [inline], [private], [inherited]
```

Definition at line 948 of file coder_array.h.

```
948         {
949             if (_newNumel > data_.capacity()) {
950                 SZ i = data_.capacity();
951                 if (i < 16) {
952                     i = 16;
953                 }
954                 while (i < _newNumel) {
955                     if (i > 1073741823) {
956                         i = MAX_int32_T;
957                     } else {
958                         i <<= 1;
959                     }
960                 }
961                 data_.reserve(i);
962             }
963             data_.resize(_newNumel);
964         }
965     }
```

6.1.4.30 index() [1/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 795 of file coder_array.h.

```
795      {
796          ::coder::detail::match_dimensions<N == 1>::check ();
797          const SZ indices[] = {_n1};
798          return ::coder::detail::index_nd<1>::compute(size_, indices);
799      }
```

6.1.4.31 index() [2/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 800 of file coder_array.h.

```
800      {
801          ::coder::detail::match_dimensions<N == 2>::check ();
802          const SZ indices[] = {_n1, _n2};
803          return ::coder::detail::index_nd<2>::compute(size_, indices);
804      }
```

6.1.4.32 index() [3/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 805 of file coder_array.h.

```
805      {
806          ::coder::detail::match_dimensions<N == 3>::check ();
807          const SZ indices[] = {_n1, _n2, _n3};
808          return ::coder::detail::index_nd<3>::compute(size_, indices);
809      }
```

6.1.4.33 index() [4/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 810 of file coder_array.h.

```
810      {
811          ::coder::detail::match_dimensions<N == 4>::check ();
812          const SZ indices[] = {_n1, _n2, _n3, _n4};
813          return ::coder::detail::index_nd<4>::compute(size_, indices);
814      }
```

6.1.4.34 index() [5/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 815 of file coder_array.h.

```
815
816     ::coder::detail::match_dimensions<N == 5>::check();
817     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5};
818     return ::coder::detail::index_nd<5>::compute(size_, indices);
819 }
```

6.1.4.35 index() [6/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 820 of file coder_array.h.

```
820
821     ::coder::detail::match_dimensions<N == 6>::check();
822     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6};
823     return ::coder::detail::index_nd<6>::compute(size_, indices);
824 }
```

6.1.4.36 index() [7/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 825 of file coder_array.h.

```
825
826     ::coder::detail::match_dimensions<N == 7>::check();
827     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
828     return ::coder::detail::index_nd<7>::compute(size_, indices);
829 }
```

6.1.4.37 index() [8/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 830 of file coder_array.h.

```
830
831     ::coder::detail::match_dimensions<N == 8>::check();
832     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
833     return ::coder::detail::index_nd<8>::compute(size_, indices);
834 }
```

6.1.4.38 index() [9/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 835 of file coder_array.h.

```
835
836     ::coder::detail::match_dimensions<N == 9>::check();
837     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
838     return ::coder::detail::index_nd<9>::compute(size_, indices);
839 }
```

6.1.4.39 index() [10/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 840 of file coder_array.h.

```
841     {
842         ::coder::detail::match_dimensions<N == 10>::check();
843         const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
844         return ::coder::detail::index_nd<10>::compute(size_, indices);
845     }
```

6.1.4.40 is_owner()

```
bool coder::array_base< T, SizeType , N >::is_owner [inline], [inherited]
```

Definition at line 587 of file coder_array.h.

```
587     {
588         return data_.is_owner();
589     }
```

6.1.4.41 numel()

```
SizeType coder::array_base< T, SizeType , N >::numel [inline], [inherited]
```

Definition at line 791 of file coder_array.h.

```
791     {
792         return ::coder::detail::numel<N>::compute(size_);
793     }
```

6.1.4.42 operator[]() [1/2]

```
T& coder::array_base< T, SizeType , N >::operator[] (
    SizeType _index ) [inline], [inherited]
```

Definition at line 763 of file coder_array.h.

```
763     {
764         return data_[_index];
765     }
```

6.1.4.43 operator[]() [2/2]

```
const T& coder::array_base< T, SizeType , N >::operator[] (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 767 of file coder_array.h.

```
767     {
768         return data_[_index];
769     }
```

6.1.4.44 reshape() [1/10]

```
array_base<T, SizeType , 1> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 710 of file coder_array.h.

```
710     {
711         const SZ ns[] = {_n1};
712         return reshape_n(ns);
713     }
```

6.1.4.45 reshape() [2/10]

```
array_base<T, SizeType , 2> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 715 of file coder_array.h.

```
715
716     const SZ ns[] = {_n1, _n2};
717     return reshape_n(ns);
718 }
```

6.1.4.46 reshape() [3/10]

```
array_base<T, SizeType , 3> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 720 of file coder_array.h.

```
720
721     const SZ ns[] = {_n1, _n2, _n3};
722     return reshape_n(ns);
723 }
```

6.1.4.47 reshape() [4/10]

```
array_base<T, SizeType , 4> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 725 of file coder_array.h.

```
725
726     const SZ ns[] = {_n1, _n2, _n3, _n4};
727     return reshape_n(ns);
728 }
```

6.1.4.48 reshape() [5/10]

```
array_base<T, SizeType , 5> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 730 of file coder_array.h.

```
730
731     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5};
732     return reshape_n(ns);
733 }
```

6.1.4.49 `reshape()` [6/10]

```
array_base<T, SizeType , 6> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 735 of file `coder_array.h`.

```
735
736     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6};
737     return reshape_n(ns);
738 }
```

6.1.4.50 `reshape()` [7/10]

```
array_base<T, SizeType , 7> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 740 of file `coder_array.h`.

```
740
741     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
742     return reshape_n(ns);
743 }
```

6.1.4.51 `reshape()` [8/10]

```
array_base<T, SizeType , 8> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 745 of file `coder_array.h`.

```
746     {
747         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
748         return reshape_n(ns);
749     }
```

6.1.4.52 reshape() [9/10]

```
array_base<T, SizeType , 9> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 752 of file coder_array.h.

```
752
753     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
754     return reshape_n(ns);
755 }
```

6.1.4.53 reshape() [10/10]

```
array_base<T, SizeType , 10> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 758 of file coder_array.h.

```
758
759     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
760     return reshape_n(ns);
761 }
```

6.1.4.54 reshape_n()

```
array_base<T, SizeType , N1> coder::array_base< T, SizeType , N >::reshape_n (
    const SizeType (&) _ns[N1] ) const [inline], [inherited]
```

Definition at line 705 of file coder_array.h.

```
705
706     array_base<T, SZ, N1> reshaped(const_cast<T*>(&data_[0]), _ns);
707     return reshaped;
708 }
```

6.1.4.55 set() [1/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1 ) [inline], [inherited]

Definition at line 481 of file coder_array.h.
481 {
482     ::coder::detail::match_dimensions<N == 1>::check ();
483     data_.set (_data, _n1);
484     size_[0] = _n1;
485 }
```

6.1.4.56 set() [2/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 487 of file coder_array.h.

```
487 {
488     ::coder::detail::match_dimensions<N == 2>::check ();
489     data_.set (_data, _n1 * _n2);
490     size_[0] = _n1;
491     size_[1] = _n2;
492 }
```

6.1.4.57 set() [3/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 494 of file coder_array.h.

```
494 {
495     ::coder::detail::match_dimensions<N == 3>::check ();
496     data_.set (_data, _n1 * _n2 * _n3);
497     size_[0] = _n1;
498     size_[1] = _n2;
499     size_[2] = _n3;
500 }
```

6.1.4.58 set() [4/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 502 of file coder_array.h.

```
502 {
503     ::coder::detail::match_dimensions<N == 4>::check ();
504     data_.set (_data, _n1 * _n2 * _n3 * _n4);
505     size_[0] = _n1;
506     size_[1] = _n2;
507     size_[2] = _n3;
508     size_[3] = _n4;
509 }
```

6.1.4.59 set() [5/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 511 of file coder_array.h.

```
511
512     ::coder::detail::match_dimensions<N == 5>::check();
513     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5);
514     size_[0] = _n1;
515     size_[1] = _n2;
516     size_[2] = _n3;
517     size_[3] = _n4;
518     size_[4] = _n5;
519 }
```

6.1.4.60 set() [6/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 521 of file coder_array.h.

```
521
522     ::coder::detail::match_dimensions<N == 6>::check();
523     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6);
524     size_[0] = _n1;
525     size_[1] = _n2;
526     size_[2] = _n3;
527     size_[3] = _n4;
528     size_[4] = _n5;
529     size_[5] = _n6;
530 }
```

6.1.4.61 set() [7/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 532 of file coder_array.h.

```

532
533     ::coder::detail::match_dimensions<N == 7>::check();
534     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7);
535     size_[0] = _n1;
536     size_[1] = _n2;
537     size_[2] = _n3;
538     size_[3] = _n4;
539     size_[4] = _n5;
540     size_[5] = _n6;
541     size_[6] = _n7;
542 }

```

6.1.4.62 set() [8/10]

```

void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]

```

Definition at line 544 of file coder_array.h.

```

544
545     ::coder::detail::match_dimensions<N == 8>::check();
546     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8);
547     size_[0] = _n1;
548     size_[1] = _n2;
549     size_[2] = _n3;
550     size_[3] = _n4;
551     size_[4] = _n5;
552     size_[5] = _n6;
553     size_[6] = _n7;
554     size_[7] = _n8;
555 }

```

6.1.4.63 set() [9/10]

```

void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]

```

Definition at line 557 of file coder_array.h.

```

557
558     ::coder::detail::match_dimensions<N == 9>::check();
559     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9);
560     size_[0] = _n1;
561     size_[1] = _n2;
562     size_[2] = _n3;
563     size_[3] = _n4;

```

```

564     size_[4] = _n5;
565     size_[5] = _n6;
566     size_[6] = _n7;
567     size_[7] = _n8;
568     size_[8] = _n9;
569 }

```

6.1.4.64 set() [10/10]

```

void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]

```

Definition at line 572 of file coder_array.h.

```

572 {
573     ::coder::detail::match_dimensions<N == 10>::check();
574     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9 * _n10);
575     size_[0] = _n1;
576     size_[1] = _n2;
577     size_[2] = _n3;
578     size_[3] = _n4;
579     size_[4] = _n5;
580     size_[5] = _n6;
581     size_[6] = _n7;
582     size_[7] = _n8;
583     size_[8] = _n9;
584     size_[9] = _n10;
585 }

```

6.1.4.65 set_owner()

```

void coder::array_base< T, SizeType , N >::set_owner (
    bool b ) [inline], [inherited]

```

Definition at line 591 of file coder_array.h.

```

591 {
592     data_.set_owner(b);
593 }

```

6.1.4.66 set_size() [1/10]

```

void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1 ) [inline], [inherited]

```

Definition at line 599 of file coder_array.h.

```

599 {
600     ::coder::detail::match_dimensions<N == 1>::check();
601     size_[0] = _n1;
602     ensureCapacity(numel());
603 }

```

6.1.4.67 set_size() [2/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 605 of file coder_array.h.

```
605     {
606         ::coder::detail::match_dimensions<N == 2>::check();
607         size_[0] = _n1;
608         size_[1] = _n2;
609         ensureCapacity(numel());
610     }
```

6.1.4.68 set_size() [3/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 612 of file coder_array.h.

```
612     {
613         ::coder::detail::match_dimensions<N == 3>::check();
614         size_[0] = _n1;
615         size_[1] = _n2;
616         size_[2] = _n3;
617         ensureCapacity(numel());
618     }
```

6.1.4.69 set_size() [4/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 620 of file coder_array.h.

```
620     {
621         ::coder::detail::match_dimensions<N == 4>::check();
622         size_[0] = _n1;
623         size_[1] = _n2;
624         size_[2] = _n3;
625         size_[3] = _n4;
626         ensureCapacity(numel());
627     }
```

6.1.4.70 set_size() [5/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 629 of file coder_array.h.

```
629
630     ::coder::detail::match_dimensions<N == 5>::check();
631     size_[0] = _n1;
632     size_[1] = _n2;
633     size_[2] = _n3;
634     size_[3] = _n4;
635     size_[4] = _n5;
636     ensureCapacity(numel());
637 }
```

6.1.4.71 set_size() [6/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 639 of file coder_array.h.

```
639
640     ::coder::detail::match_dimensions<N == 6>::check();
641     size_[0] = _n1;
642     size_[1] = _n2;
643     size_[2] = _n3;
644     size_[3] = _n4;
645     size_[4] = _n5;
646     size_[5] = _n6;
647     ensureCapacity(numel());
648 }
```

6.1.4.72 set_size() [7/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 650 of file coder_array.h.

```
650
651     ::coder::detail::match_dimensions<N == 7>::check();
652     size_[0] = _n1;
653     size_[1] = _n2;
654     size_[2] = _n3;
655     size_[3] = _n4;
656     size_[4] = _n5;
657     size_[5] = _n6;
658     size_[6] = _n7;
659     ensureCapacity(numel());
660 }
```

6.1.4.73 set_size() [8/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 662 of file coder_array.h.

```
662
663     ::coder::detail::match_dimensions<N == 8>::check();
664     size_[0] = _n1;
665     size_[1] = _n2;
666     size_[2] = _n3;
667     size_[3] = _n4;
668     size_[4] = _n5;
669     size_[5] = _n6;
670     size_[6] = _n7;
671     size_[7] = _n8;
672     ensureCapacity(numel());
673 }
```

6.1.4.74 set_size() [9/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]
```

Definition at line 675 of file coder_array.h.

```
675
676     ::coder::detail::match_dimensions<N == 9>::check();
677     size_[0] = _n1;
678     size_[1] = _n2;
679     size_[2] = _n3;
680     size_[3] = _n4;
681     size_[4] = _n5;
682     size_[5] = _n6;
683     size_[6] = _n7;
684     size_[7] = _n8;
685     size_[8] = _n9;
686     ensureCapacity(numel());
687 }
```

6.1.4.75 set_size() [10/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]
```

Definition at line 689 of file coder_array.h.

```
689
690     ::coder::detail::match_dimensions<N == 10>::check();
691     size_[0] = _n1;
692     size_[1] = _n2;
693     size_[2] = _n3;
694     size_[3] = _n4;
695     size_[4] = _n5;
696     size_[5] = _n6;
697     size_[6] = _n7;
698     size_[7] = _n8;
699     size_[8] = _n9;
700     size_[9] = _n10;
701     ensureCapacity(numel());
702 }
```

6.1.4.76 size() [1/2]

```
const SizeType * coder::array_base< T, SizeType , N >::size [inline], [inherited]
```

Definition at line 783 of file coder_array.h.

```
783
784     return &size_[0];
785 }
```

6.1.4.77 size() [2/2]

```
SizeType coder::array_base< T, SizeType , N >::size (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 787 of file coder_array.h.

```
787
788     return size_[_index];
789 }
```

6.1.5 Member Data Documentation

6.1.5.1 `data_`

```
::coder::detail::data_ptr<T, SizeType > coder::array_base< T, SizeType , N >::data_ [protected],  
[inherited]
```

Definition at line 944 of file `coder_array.h`.

6.1.5.2 `size_`

```
SizeType coder::array_base< T, SizeType , N >::size_[N] [protected], [inherited]
```

Definition at line 945 of file `coder_array.h`.

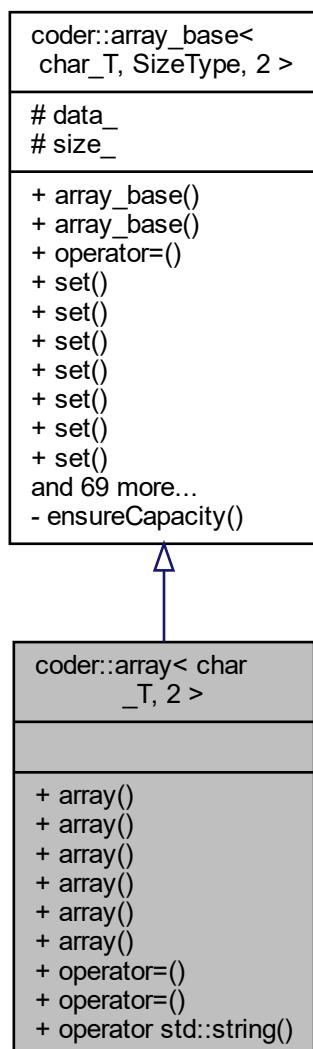
The documentation for this class was generated from the following file:

- codegen/lib(Func_fovPathPlanning/[coder_array.h](#)

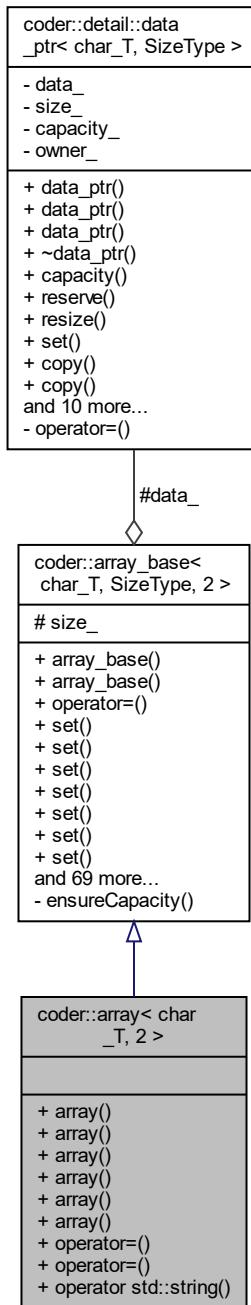
6.2 `coder::array< char_T, 2 >` Class Reference

```
#include <coder_array.h>
```

Inheritance diagram for coder::array< char_T, 2 >:



Collaboration diagram for coder::array< char_T, 2 >:



Public Types

- `typedef char_T value_type`
 - `typedef SizeType size_type`

Public Member Functions

- `array ()`

- `array (const array< char_T, 2 > &_other)`
- `array (const Base &_other)`
- `array (const std::string &_str)`
- `array (const char_T *_str)`
- `array (const std::vector< char_T > &_vec)`
- `array & operator= (const std::string &_str)`
- `array & operator= (const char_T *_str)`
- `operator std::string () const`
- `void set (char_T *_data, SizeType _n1)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set (char_T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `bool is_owner () const`
- `void set_owner (bool b)`
- `SizeType capacity () const`
- `void set_size (SizeType _n1)`
- `void set_size (SizeType _n1, SizeType _n2)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `array_base< char_T, SizeType, N1 > reshape_n (const SizeType(&_ns)[N1]) const`
- `array_base< char_T, SizeType, 1 > reshape (SizeType _n1) const`
- `array_base< char_T, SizeType, 2 > reshape (SizeType _n1, SizeType _n2) const`
- `array_base< char_T, SizeType, 3 > reshape (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `array_base< char_T, SizeType, 4 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `array_base< char_T, SizeType, 5 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `array_base< char_T, SizeType, 6 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `array_base< char_T, SizeType, 7 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `array_base< char_T, SizeType, 8 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `array_base< char_T, SizeType, 9 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`

- `array_base< char_T, SizeType, 10 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- `char_T & operator[] (SizeType _index)`
- `const char_T & operator[] (SizeType _index) const`
- `void clear ()`
- `char_T * data ()`
- `const char_T * data () const`
- `const SizeType * size () const`
- `SizeType size (SizeType _index) const`
- `SizeType numel () const`
- `SizeType index (SizeType _n1) const`
- `SizeType index (SizeType _n1, SizeType _n2) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- `char_T & at (SizeType _i1)`
- `char_T & at (SizeType _i1, SizeType _i2)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9)`
- `char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10)`
- `const char_T & at (SizeType _i1) const`
- `const char_T & at (SizeType _i1, SizeType _i2) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9) const`
- `const char_T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10) const`
- `array_iterator< array_base< char_T, SizeType, N > > begin ()`
- `const_array_iterator< array_base< char_T, SizeType, N > > begin () const`
- `array_iterator< array_base< char_T, SizeType, N > > end ()`
- `const_array_iterator< array_base< char_T, SizeType, N > > end () const`

Protected Attributes

- ::coder::detail::data_ptr< char_T, SizeType > data_
- SizeType size_[N]

Private Types

- typedef array_base< char_T, SizeType, 2 > Base

Private Member Functions

- void ensureCapacity (SizeType _newNumel)

6.2.1 Detailed Description

Definition at line 991 of file coder_array.h.

6.2.2 Member Typedef Documentation

6.2.2.1 Base

```
typedef array_base<char_T, SizeType, 2> coder::array< char_T, 2 >::Base [private]
```

Definition at line 993 of file coder_array.h.

6.2.2.2 size_type

```
typedef SizeType coder::array_base< char_T , SizeType , N >::size_type [inherited]
```

Definition at line 464 of file coder_array.h.

6.2.2.3 value_type

```
typedef char_T coder::array_base< char_T , SizeType , N >::value_type [inherited]
```

Definition at line 463 of file coder_array.h.

6.2.3 Constructor & Destructor Documentation

6.2.3.1 array() [1/6]

```
coder::array< char_T, 2 >::array ( ) [inline]
```

Definition at line 996 of file coder_array.h.
 997 : array_base() {
 998 }

6.2.3.2 array() [2/6]

```
coder::array< char_T, 2 >::array ( const array< char_T, 2 > & _other ) [inline]
```

Definition at line 999 of file coder_array.h.
 1000 : Base(_other) {
 1001 }

6.2.3.3 array() [3/6]

```
coder::array< char_T, 2 >::array ( const Base & _other ) [inline]
```

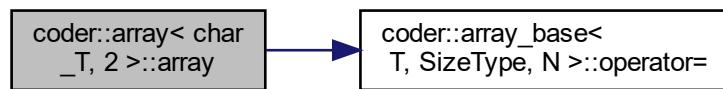
Definition at line 1002 of file coder_array.h.
 1003 : Base(_other) {
 1004 }

6.2.3.4 array() [4/6]

```
coder::array< char_T, 2 >::array ( const std::string & _str ) [inline]
```

Definition at line 1006 of file coder_array.h.
 1006 {
 1007 operator=(_str);
 1008 }

Here is the call graph for this function:



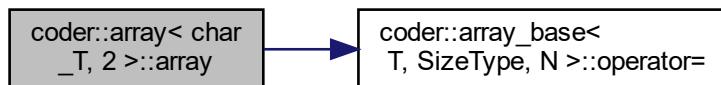
6.2.3.5 array() [5/6]

```
coder::array< char_T, 2 >::array (
    const char_T * _str )  [inline]
```

Definition at line 1010 of file coder_array.h.

```
1010
1011     operator=(_str);
1012 }
```

Here is the call graph for this function:



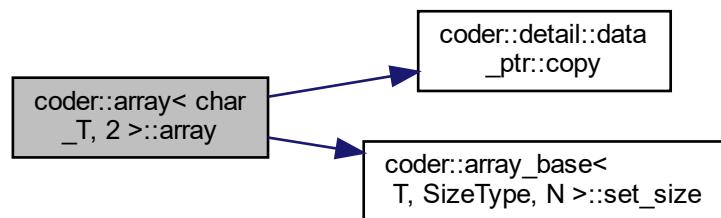
6.2.3.6 array() [6/6]

```
coder::array< char_T, 2 >::array (
    const std::vector< char_T > & _vec )  [inline]
```

Definition at line 1014 of file coder_array.h.

```
1014
1015     SizeType n = static_cast<SizeType>(_vec.size());
1016     set_size(1, n);
1017     data_.copy(&_vec[0], n);
1018 }
```

Here is the call graph for this function:



6.2.4 Member Function Documentation

6.2.4.1 at() [1/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1 ) [inline], [inherited]
```

Definition at line 847 of file coder_array.h.

```
847     {
848         ::coder::detail::match_dimensions<N == 1>::check ();
849         return data_[_i1];
850     }
```

6.2.4.2 at() [2/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1 ) const [inline], [inherited]
```

Definition at line 888 of file coder_array.h.

```
888     {
889         ::coder::detail::match_dimensions<N == 1>::check ();
890         return data_[_i1];
891     }
```

6.2.4.3 at() [3/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) [inline], [inherited]
```

Definition at line 851 of file coder_array.h.

```
851     {
852         ::coder::detail::match_dimensions<N == 2>::check ();
853         return data_[index(_i1, _i2)];
854     }
```

6.2.4.4 at() [4/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) const [inline], [inherited]
```

Definition at line 892 of file coder_array.h.

```
892     {
893         ::coder::detail::match_dimensions<N == 2>::check ();
894         return data_[index(_i1, _i2)];
895     }
```

6.2.4.5 at() [5/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) [inline], [inherited]
```

Definition at line 855 of file coder_array.h.

```
855     {
856         ::coder::detail::match_dimensions<N == 3>::check();
857         return data_[index(_i1, _i2, _i3)];
858     }
```

6.2.4.6 at() [6/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) const [inline], [inherited]
```

Definition at line 896 of file coder_array.h.

```
896     {
897         ::coder::detail::match_dimensions<N == 3>::check();
898         return data_[index(_i1, _i2, _i3)];
899     }
```

6.2.4.7 at() [7/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) [inline], [inherited]
```

Definition at line 859 of file coder_array.h.

```
859     {
860         ::coder::detail::match_dimensions<N == 4>::check();
861         return data_[index(_i1, _i2, _i3, _i4)];
862     }
```

6.2.4.8 at() [8/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) const [inline], [inherited]
```

Definition at line 900 of file coder_array.h.

```
900     {
901         ::coder::detail::match_dimensions<N == 4>::check();
902         return data_[index(_i1, _i2, _i3, _i4)];
903     }
```

6.2.4.9 at() [9/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) [inline], [inherited]
```

Definition at line 863 of file coder_array.h.

```
863
864     ::coder::detail::match_dimensions<N == 5>::check();
865     return data_[index(_i1, _i2, _i3, _i4, _i5)];
866 }
```

6.2.4.10 at() [10/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) const [inline], [inherited]
```

Definition at line 904 of file coder_array.h.

```
904
905     ::coder::detail::match_dimensions<N == 5>::check();
906     return data_[index(_i1, _i2, _i3, _i4, _i5)];
907 }
```

6.2.4.11 at() [11/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) [inline], [inherited]
```

Definition at line 867 of file coder_array.h.

```
867
868     ::coder::detail::match_dimensions<N == 6>::check();
869     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
870 }
```

6.2.4.12 at() [12/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) const [inline], [inherited]
```

Definition at line 908 of file coder_array.h.

```
908
909     ::coder::detail::match_dimensions<N == 6>::check();
910     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
911 }
```

6.2.4.13 at() [13/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) [inline], [inherited]
```

Definition at line 871 of file coder_array.h.

```
871
872     ::coder::detail::match_dimensions<N == 7>::check();
873     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
874 }
```

6.2.4.14 at() [14/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) const [inline], [inherited]
```

Definition at line 912 of file coder_array.h.

```
912
913     ::coder::detail::match_dimensions<N == 7>::check();
914     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
915 }
```

6.2.4.15 at() [15/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) [inline], [inherited]
```

Definition at line 875 of file coder_array.h.

```
875
876     ::coder::detail::match_dimensions<N == 8>::check();
877     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
878 }
```

6.2.4.16 at() [16/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) const [inline], [inherited]
```

Definition at line 916 of file coder_array.h.

```
916
917     ::coder::detail::match_dimensions<N == 8>::check();
918     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
919 }
```

6.2.4.17 at() [17/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) [inline], [inherited]
```

Definition at line 879 of file coder_array.h.

```
879
880     ::coder::detail::match_dimensions<N == 9>::check();
881     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
882 }
```

6.2.4.18 at() [18/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) const [inline], [inherited]
```

Definition at line 920 of file coder_array.h.

```
920 {
921     ::coder::detail::match_dimensions<N == 9>::check();
922     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
923 }
```

6.2.4.19 at() [19/20]

```
char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) [inline], [inherited]
```

Definition at line 883 of file coder_array.h.

```
883 {
884     ::coder::detail::match_dimensions<N == 10>::check();
885     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
886 }
```

6.2.4.20 at() [20/20]

```
const char_T & coder::array_base< char_T , SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) const [inline], [inherited]
```

Definition at line 924 of file coder_array.h.

```
925 {
926     ::coder::detail::match_dimensions<N == 10>::check();
927     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
928 }
```

6.2.4.21 begin() [1/2]

```
array_iterator<array_base<char_T , SizeType , N> > coder::array_base< char_T , SizeType , N
>::begin [inline], [inherited]
```

Definition at line 930 of file coder_array.h.

```
930
931     return array_iterator<array_base<T, SZ, N> >(this, 0);
932 }
```

6.2.4.22 begin() [2/2]

```
const_array_iterator<array_base<char_T , SizeType , N> > coder::array_base< char_T , SizeType
, N >::begin [inline], [inherited]
```

Definition at line 936 of file coder_array.h.

```
936
937     return const_array_iterator<array_base<T, SZ, N> >(this, 0);
938 }
```

6.2.4.23 capacity()

```
SizeType coder::array_base< char_T , SizeType , N >::capacity [inline], [inherited]
```

Definition at line 595 of file coder_array.h.

```
595
596     return data_.capacity();
597 }
```

6.2.4.24 clear()

```
void coder::array_base< char_T , SizeType , N >::clear [inline], [inherited]
```

Definition at line 771 of file coder_array.h.

```
771
772     data_.clear();
773 }
```

6.2.4.25 data() [1/2]

```
char_T * coder::array_base< char_T , SizeType , N >::data [inline], [inherited]
```

Definition at line 775 of file coder_array.h.

```
775
776     return data_;
777 }
```

6.2.4.26 data() [2/2]

```
const char_T * coder::array_base< char_T , SizeType , N >::data [inline], [inherited]
```

Definition at line 779 of file coder_array.h.

```
779         {
780             return data_;
781         }
```

6.2.4.27 end() [1/2]

```
array_iterator<array_base<char_T , SizeType , N> > coder::array_base< char_T , SizeType , N >::end [inline], [inherited]
```

Definition at line 933 of file coder_array.h.

```
933         {
934             return array_iterator<array_base<T, SZ, N> >(this, this->numel());
935         }
```

6.2.4.28 end() [2/2]

```
const_array_iterator<array_base<char_T , SizeType , N> > coder::array_base< char_T , SizeType , N >::end [inline], [inherited]
```

Definition at line 939 of file coder_array.h.

```
939         {
940             return const_array_iterator<array_base<T, SZ, N> >(this, this->numel());
941         }
```

6.2.4.29 ensureCapacity()

```
void coder::array_base< char_T , SizeType , N >::ensureCapacity (
    SizeType _newNumel ) [inline], [private], [inherited]
```

Definition at line 948 of file coder_array.h.

```
948         {
949             if (_newNumel > data_.capacity()) {
950                 SZ i = data_.capacity();
951                 if (i < 16) {
952                     i = 16;
953                 }
954                 while (i < _newNumel) {
955                     if (i > 1073741823) {
956                         i = MAX_int32_T;
957                     } else {
958                         i <<= 1;
959                     }
960                 }
961                 data_.reserve(i);
962             }
963             data_.resize(_newNumel);
964         }
965     }
```

6.2.4.30 index() [1/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 795 of file coder_array.h.

```
795      {
796          ::coder::detail::match_dimensions<N == 1>::check ();
797          const SZ indices[] = {_n1};
798          return ::coder::detail::index_nd<1>::compute(size_, indices);
799      }
```

6.2.4.31 index() [2/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 800 of file coder_array.h.

```
800      {
801          ::coder::detail::match_dimensions<N == 2>::check ();
802          const SZ indices[] = {_n1, _n2};
803          return ::coder::detail::index_nd<2>::compute(size_, indices);
804      }
```

6.2.4.32 index() [3/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 805 of file coder_array.h.

```
805      {
806          ::coder::detail::match_dimensions<N == 3>::check ();
807          const SZ indices[] = {_n1, _n2, _n3};
808          return ::coder::detail::index_nd<3>::compute(size_, indices);
809      }
```

6.2.4.33 index() [4/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 810 of file coder_array.h.

```
810      {
811          ::coder::detail::match_dimensions<N == 4>::check ();
812          const SZ indices[] = {_n1, _n2, _n3, _n4};
813          return ::coder::detail::index_nd<4>::compute(size_, indices);
814      }
```

6.2.4.34 index() [5/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 815 of file coder_array.h.

```
815
816     ::coder::detail::match_dimensions<N == 5>::check();
817     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5};
818     return ::coder::detail::index_nd<5>::compute(size_, indices);
819 }
```

6.2.4.35 index() [6/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 820 of file coder_array.h.

```
820
821     ::coder::detail::match_dimensions<N == 6>::check();
822     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6};
823     return ::coder::detail::index_nd<6>::compute(size_, indices);
824 }
```

6.2.4.36 index() [7/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 825 of file coder_array.h.

```
825
826     ::coder::detail::match_dimensions<N == 7>::check();
827     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
828     return ::coder::detail::index_nd<7>::compute(size_, indices);
829 }
```

6.2.4.37 index() [8/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 830 of file coder_array.h.

```
830
831     ::coder::detail::match_dimensions<N == 8>::check();
832     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
833     return ::coder::detail::index_nd<8>::compute(size_, indices);
834 }
```

6.2.4.38 index() [9/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 835 of file coder_array.h.

```
835
836     ::coder::detail::match_dimensions<N == 9>::check();
837     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
838     return ::coder::detail::index_nd<9>::compute(size_, indices);
839 }
```

6.2.4.39 index() [10/10]

```
SizeType coder::array_base< char_T , SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 840 of file coder_array.h.

```
841
842     ::coder::detail::match_dimensions<N == 10>::check();
843     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
844     return ::coder::detail::index_nd<10>::compute(size_, indices);
845 }
```

6.2.4.40 is_owner()

```
bool coder::array_base< char_T , SizeType , N >::is_owner [inline], [inherited]
```

Definition at line 587 of file coder_array.h.

```
587         {
588             return data_.is_owner();
589         }
```

6.2.4.41 numel()

```
SizeType coder::array_base< char_T , SizeType , N >::numel [inline], [inherited]
```

Definition at line 791 of file coder_array.h.

```
791         {
792             return ::coder::detail::numel<N>::compute(size_);
793         }
```

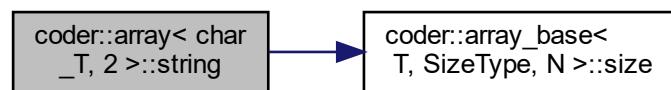
6.2.4.42 operator std::string()

```
coder::array< char_T, 2 >::operator std::string () const [inline]
```

Definition at line 1034 of file coder_array.h.

```
1034         {
1035             return std::string(static_cast<const char*>(&(*this)[0]), static_cast<int>(size(1)));
1036         }
```

Here is the call graph for this function:



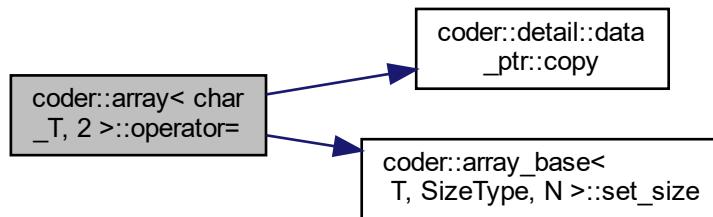
6.2.4.43 operator=() [1/2]

```
array& coder::array< char_T, 2 >::operator= (
    const char_T * _str )  [inline]
```

Definition at line 1027 of file coder_array.h.

```
1027      {
1028          SizeType n = static_cast<SizeType>(strlen(_str));
1029          set_size(1, n);
1030          data_.copy(_str, n);
1031          return *this;
1032      }
```

Here is the call graph for this function:



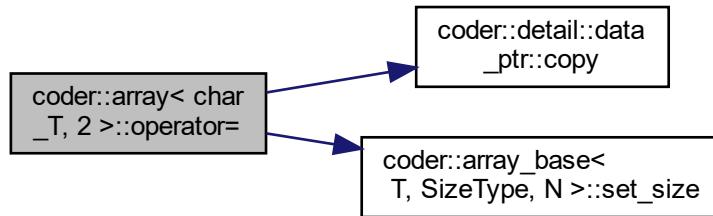
6.2.4.44 operator=() [2/2]

```
array& coder::array< char_T, 2 >::operator= (
    const std::string & _str )  [inline]
```

Definition at line 1020 of file coder_array.h.

```
1020      {
1021          SizeType n = static_cast<SizeType>(_str.size());
1022          set_size(1, n);
1023          data_.copy(_str.c_str(), n);
1024          return *this;
1025      }
```

Here is the call graph for this function:



6.2.4.45 operator[]() [1/2]

```
char_T & coder::array_base< char_T , SizeType , N >::operator[] (
    SizeType _index ) [inline], [inherited]
```

Definition at line 763 of file coder_array.h.

```
763     {
764         return data_[_index];
765     }
```

6.2.4.46 operator[]() [2/2]

```
const char_T & coder::array_base< char_T , SizeType , N >::operator[] (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 767 of file coder_array.h.

```
767     {
768         return data_[_index];
769     }
```

6.2.4.47 reshape() [1/10]

```
array_base<char_T , SizeType , 1> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 710 of file coder_array.h.

```
710     {
711         const SZ ns[] = {_n1};
712         return reshape_n(ns);
713     }
```

6.2.4.48 reshape() [2/10]

```
array_base<char_T , SizeType , 2> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 715 of file coder_array.h.

```
715     {
716         const SZ ns[] = {_n1, _n2};
717         return reshape_n(ns);
718     }
```

6.2.4.49 `reshape()` [3/10]

```
array_base<char_T , SizeType , 3> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 720 of file `coder_array.h`.

```
720
721     const SZ ns[] = {_n1, _n2, _n3};
722     return reshape_n(ns);
723 }
```

6.2.4.50 `reshape()` [4/10]

```
array_base<char_T , SizeType , 4> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 725 of file `coder_array.h`.

```
725
726     const SZ ns[] = {_n1, _n2, _n3, _n4};
727     return reshape_n(ns);
728 }
```

6.2.4.51 `reshape()` [5/10]

```
array_base<char_T , SizeType , 5> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 730 of file `coder_array.h`.

```
730
731     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5};
732     return reshape_n(ns);
733 }
```

6.2.4.52 `reshape()` [6/10]

```
array_base<char_T , SizeType , 6> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 735 of file `coder_array.h`.

```
735
736     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6};
737     return reshape_n(ns);
738 }
```

6.2.4.53 reshape() [7/10]

```
array_base<char_T , SizeType , 7> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 740 of file coder_array.h.

```
740
741     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
742     return reshape_n(ns);
743 }
```

6.2.4.54 reshape() [8/10]

```
array_base<char_T , SizeType , 8> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 745 of file coder_array.h.

```
746
747     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
748     return reshape_n(ns);
749 }
```

6.2.4.55 reshape() [9/10]

```
array_base<char_T , SizeType , 9> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 752 of file coder_array.h.

```
752
753     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
754     return reshape_n(ns);
755 }
```

6.2.4.56 `reshape()` [10/10]

```
array_base<char_T , SizeType , 10> coder::array_base< char_T , SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 758 of file `coder_array.h`.

```
758
759     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
760     return reshape_n(ns);
761 }
```

6.2.4.57 `reshape_n()`

```
array_base<char_T , SizeType , N1> coder::array_base< char_T , SizeType , N >::reshape_n (
    const SizeType (&) _ns[N1] ) const [inline], [inherited]
```

Definition at line 705 of file `coder_array.h`.

```
705
706     array_base<T, SZ, N1> reshaped(const_cast<T*>(&data_[0]), _ns);
707     return reshaped;
708 }
```

6.2.4.58 `set()` [1/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1 ) [inline], [inherited]
```

Definition at line 481 of file `coder_array.h`.

```
481
482     ::coder::detail::match_dimensions<N == 1>::check();
483     data_.set(_data, _n1);
484     size_[0] = _n1;
485 }
```

6.2.4.59 `set()` [2/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 487 of file `coder_array.h`.

```
487
488     ::coder::detail::match_dimensions<N == 2>::check();
489     data_.set(_data, _n1 * _n2);
490     size_[0] = _n1;
491     size_[1] = _n2;
492 }
```

6.2.4.60 set() [3/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 494 of file coder_array.h.

```
494     {
495         ::coder::detail::match_dimensions<N == 3>::check();
496         data_.set(_data, _n1 * _n2 * _n3);
497         size_[0] = _n1;
498         size_[1] = _n2;
499         size_[2] = _n3;
500     }
```

6.2.4.61 set() [4/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 502 of file coder_array.h.

```
502     {
503         ::coder::detail::match_dimensions<N == 4>::check();
504         data_.set(_data, _n1 * _n2 * _n3 * _n4);
505         size_[0] = _n1;
506         size_[1] = _n2;
507         size_[2] = _n3;
508         size_[3] = _n4;
509     }
```

6.2.4.62 set() [5/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 511 of file coder_array.h.

```
511     {
512         ::coder::detail::match_dimensions<N == 5>::check();
513         data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5);
514         size_[0] = _n1;
515         size_[1] = _n2;
516         size_[2] = _n3;
517         size_[3] = _n4;
518         size_[4] = _n5;
519     }
```

6.2.4.63 set() [6/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 521 of file coder_array.h.

```
521
522     ::coder::detail::match_dimensions<N == 6>::check();
523     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6);
524     size_[0] = _n1;
525     size_[1] = _n2;
526     size_[2] = _n3;
527     size_[3] = _n4;
528     size_[4] = _n5;
529     size_[5] = _n6;
530 }
```

6.2.4.64 set() [7/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 532 of file coder_array.h.

```
532
533     ::coder::detail::match_dimensions<N == 7>::check();
534     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7);
535     size_[0] = _n1;
536     size_[1] = _n2;
537     size_[2] = _n3;
538     size_[3] = _n4;
539     size_[4] = _n5;
540     size_[5] = _n6;
541     size_[6] = _n7;
542 }
```

6.2.4.65 set() [8/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
```

```
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 544 of file coder_array.h.

```
544
545     ::coder::detail::match_dimensions<N == 8>::check();
546     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8);
547     size_[0] = _n1;
548     size_[1] = _n2;
549     size_[2] = _n3;
550     size_[3] = _n4;
551     size_[4] = _n5;
552     size_[5] = _n6;
553     size_[6] = _n7;
554     size_[7] = _n8;
555 }
```

6.2.4.66 set() [9/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]
```

Definition at line 557 of file coder_array.h.

```
557
558     ::coder::detail::match_dimensions<N == 9>::check();
559     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9);
560     size_[0] = _n1;
561     size_[1] = _n2;
562     size_[2] = _n3;
563     size_[3] = _n4;
564     size_[4] = _n5;
565     size_[5] = _n6;
566     size_[6] = _n7;
567     size_[7] = _n8;
568     size_[8] = _n9;
569 }
```

6.2.4.67 set() [10/10]

```
void coder::array_base< char_T , SizeType , N >::set (
    char_T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
```

```
SizeType _n9,
SizeType _n10 ) [inline], [inherited]
```

Definition at line 572 of file coder_array.h.

```
572
573     ::coder::detail::match_dimensions<N == 10>::check();
574     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9 * _n10);
575     size_[0] = _n1;
576     size_[1] = _n2;
577     size_[2] = _n3;
578     size_[3] = _n4;
579     size_[4] = _n5;
580     size_[5] = _n6;
581     size_[6] = _n7;
582     size_[7] = _n8;
583     size_[8] = _n9;
584     size_[9] = _n10;
585 }
```

{

6.2.4.68 set_owner()

```
void coder::array_base< char_T , SizeType , N >::set_owner (
    bool b ) [inline], [inherited]
```

Definition at line 591 of file coder_array.h.

```
591 {
592     data_.set_owner(b);
593 }
```

6.2.4.69 set_size() [1/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1 ) [inline], [inherited]
```

Definition at line 599 of file coder_array.h.

```
599 {
600     ::coder::detail::match_dimensions<N == 1>::check();
601     size_[0] = _n1;
602     ensureCapacity(numel());
603 }
```

6.2.4.70 set_size() [2/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 605 of file coder_array.h.

```
605 {
606     ::coder::detail::match_dimensions<N == 2>::check();
607     size_[0] = _n1;
608     size_[1] = _n2;
609     ensureCapacity(numel());
610 }
```

6.2.4.71 set_size() [3/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 612 of file coder_array.h.

```
612             {
613     ::coder::detail::match_dimensions<N == 3>::check ();
614     size_[0] = _n1;
615     size_[1] = _n2;
616     size_[2] = _n3;
617     ensureCapacity(numel ());
618 }
```

6.2.4.72 set_size() [4/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 620 of file coder_array.h.

```
620             {
621     ::coder::detail::match_dimensions<N == 4>::check ();
622     size_[0] = _n1;
623     size_[1] = _n2;
624     size_[2] = _n3;
625     size_[3] = _n4;
626     ensureCapacity(numel ());
627 }
```

6.2.4.73 set_size() [5/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 629 of file coder_array.h.

```
629             {
630     ::coder::detail::match_dimensions<N == 5>::check ();
631     size_[0] = _n1;
632     size_[1] = _n2;
633     size_[2] = _n3;
634     size_[3] = _n4;
635     size_[4] = _n5;
636     ensureCapacity(numel ());
637 }
```

6.2.4.74 set_size() [6/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 639 of file coder_array.h.

```
639
640     ::coder::detail::match_dimensions<N == 6>::check ();
641     size_[0] = _n1;
642     size_[1] = _n2;
643     size_[2] = _n3;
644     size_[3] = _n4;
645     size_[4] = _n5;
646     size_[5] = _n6;
647     ensureCapacity(numel ());
648 }
```

6.2.4.75 set_size() [7/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 650 of file coder_array.h.

```
650
651     ::coder::detail::match_dimensions<N == 7>::check ();
652     size_[0] = _n1;
653     size_[1] = _n2;
654     size_[2] = _n3;
655     size_[3] = _n4;
656     size_[4] = _n5;
657     size_[5] = _n6;
658     size_[6] = _n7;
659     ensureCapacity(numel ());
660 }
```

6.2.4.76 set_size() [8/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 662 of file coder_array.h.

```

662
663     ::coder::detail::match_dimensions<N == 8>::check();
664     size_[0] = _n1;
665     size_[1] = _n2;
666     size_[2] = _n3;
667     size_[3] = _n4;
668     size_[4] = _n5;
669     size_[5] = _n6;
670     size_[6] = _n7;
671     size_[7] = _n8;
672     ensureCapacity(numel());
673 }
```

6.2.4.77 set_size() [9/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]
```

Definition at line 675 of file coder_array.h.

```

675
676     ::coder::detail::match_dimensions<N == 9>::check();
677     size_[0] = _n1;
678     size_[1] = _n2;
679     size_[2] = _n3;
680     size_[3] = _n4;
681     size_[4] = _n5;
682     size_[5] = _n6;
683     size_[6] = _n7;
684     size_[7] = _n8;
685     size_[8] = _n9;
686     ensureCapacity(numel());
687 }
```

6.2.4.78 set_size() [10/10]

```
void coder::array_base< char_T , SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]
```

Definition at line 689 of file coder_array.h.

```

689
690     ::coder::detail::match_dimensions<N == 10>::check();
691     size_[0] = _n1;
692     size_[1] = _n2;
```

```

693     size_[2] = _n3;
694     size_[3] = _n4;
695     size_[4] = _n5;
696     size_[5] = _n6;
697     size_[6] = _n7;
698     size_[7] = _n8;
699     size_[8] = _n9;
700     size_[9] = _n10;
701     ensureCapacity(numel());
702 }

```

6.2.4.79 `size()` [1/2]

```
const SizeType * coder::array_base< char_T , SizeType , N >::size [inline], [inherited]
```

Definition at line 783 of file coder_array.h.

```

783 {
784     return &size_[0];
785 }

```

6.2.4.80 `size()` [2/2]

```
SizeType coder::array_base< char_T , SizeType , N >::size (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 787 of file coder_array.h.

```

787 {
788     return size_[_index];
789 }

```

6.2.5 Member Data Documentation

6.2.5.1 `data_`

```
::coder::detail::data_ptr<char_T , SizeType > coder::array_base< char_T , SizeType , N >::data_
    [protected], [inherited]
```

Definition at line 944 of file coder_array.h.

6.2.5.2 `size_`

```
SizeType coder::array_base< char_T , SizeType , N >::size_[N] [protected], [inherited]
```

Definition at line 945 of file coder_array.h.

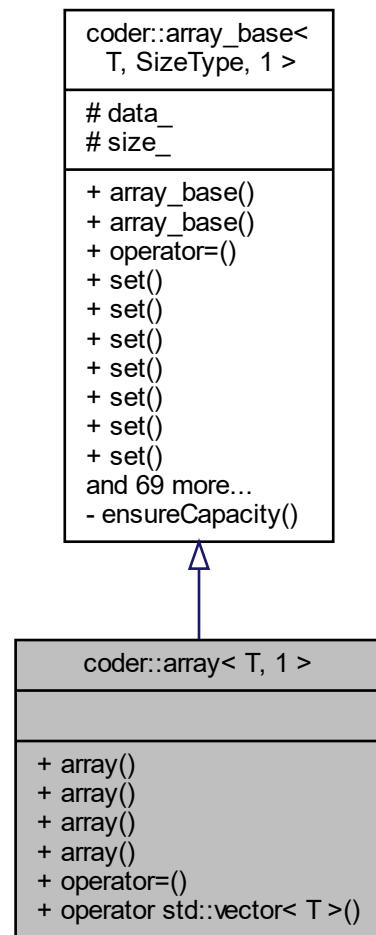
The documentation for this class was generated from the following file:

- codegen/lib(Func_fovPathPlanning/coder_array.h)

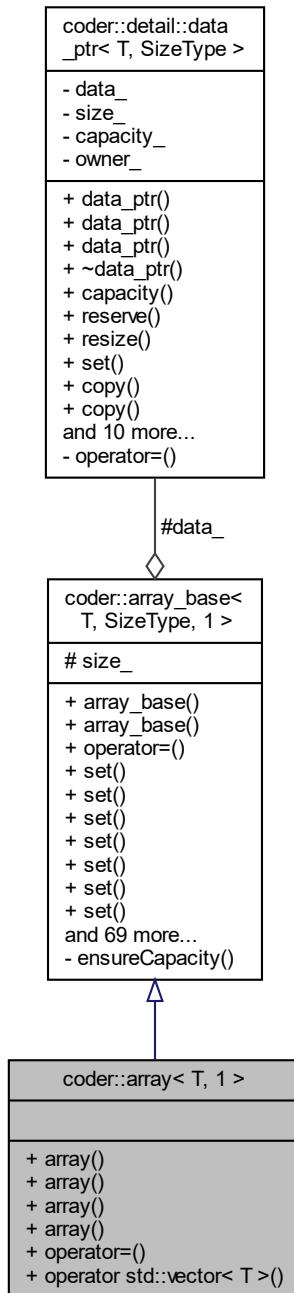
6.3 coder::array< T, 1 > Class Template Reference

```
#include <coder_array.h>
```

Inheritance diagram for coder::array< T, 1 >:



Collaboration diagram for coder::array< T, 1 >:



Public Types

- `typedef T value_type`
- `typedef SizeType size_type`

Public Member Functions

- `array ()`

- `array (const array< T, 1 > &_other)`
- `array (const Base &_other)`
- `array (const std::vector< T > &_vec)`
- `array & operator= (const std::vector< T > &_vec)`
- `operator std::vector< T > () const`
- `void set (T *_data, SizeType _n1)`
- `void set (T *_data, SizeType _n1, SizeType _n2)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `bool is_owner () const`
- `void set_owner (bool b)`
- `SizeType capacity () const`
- `void set_size (SizeType _n1)`
- `void set_size (SizeType _n1, SizeType _n2)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `array_base< T, SizeType, N1 > reshape_n (const SizeType(&_ns)[N1]) const`
- `array_base< T, SizeType, 1 > reshape (SizeType _n1) const`
- `array_base< T, SizeType, 2 > reshape (SizeType _n1, SizeType _n2) const`
- `array_base< T, SizeType, 3 > reshape (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `array_base< T, SizeType, 4 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `array_base< T, SizeType, 5 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `array_base< T, SizeType, 6 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `array_base< T, SizeType, 7 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `array_base< T, SizeType, 8 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `array_base< T, SizeType, 9 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`
- `array_base< T, SizeType, 10 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- `T & operator[] (SizeType _index)`
- `const T & operator[] (SizeType _index) const`

- void `clear ()`
- T * `data ()`
- const T * `data () const`
- const `SizeType * size () const`
- `SizeType size (SizeType _index) const`
- `SizeType numel () const`
- `SizeType index (SizeType _n1) const`
- `SizeType index (SizeType _n1, SizeType _n2) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- T & `at (SizeType _i1)`
- T & `at (SizeType _i1, SizeType _i2)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9)`
- T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10)`
- const T & `at (SizeType _i1) const`
- const T & `at (SizeType _i1, SizeType _i2) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9) const`
- const T & `at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10) const`
- `array_iterator< array_base< T, SizeType, N > > begin ()`
- `const_array_iterator< array_base< T, SizeType, N > > begin () const`
- `array_iterator< array_base< T, SizeType, N > > end ()`
- `const_array_iterator< array_base< T, SizeType, N > > end () const`

Protected Attributes

- `::coder::detail::data_ptr< T, SizeType > data_`
- `SizeType size_[N]`

Private Types

- `typedef array_base< T, SizeType, 1 > Base`

Private Member Functions

- `void ensureCapacity (SizeType _newNumel)`

6.3.1 Detailed Description

```
template<typename T>
class coder::array< T, 1 >
```

Definition at line 1076 of file coder_array.h.

6.3.2 Member Typedef Documentation

6.3.2.1 Base

```
template<typename T >
typedef array_base<T, SizeType, 1> coder::array< T, 1 >::Base [private]
```

Definition at line 1078 of file coder_array.h.

6.3.2.2 size_type

```
typedef SizeType coder::array_base< T, SizeType , N >::size_type [inherited]
```

Definition at line 464 of file coder_array.h.

6.3.2.3 value_type

```
typedef T coder::array_base< T, SizeType , N >::value_type [inherited]
```

Definition at line 463 of file coder_array.h.

6.3.3 Constructor & Destructor Documentation

6.3.3.1 array() [1/4]

```
template<typename T >
coder::array< T, 1 >::array ( ) [inline]

Definition at line 1081 of file coder_array.h.
1082     : Base() {
1083 }
```

6.3.3.2 array() [2/4]

```
template<typename T >
coder::array< T, 1 >::array (
    const array< T, 1 > & _other ) [inline]

Definition at line 1084 of file coder_array.h.
1085     : Base(_other) {
1086 }
```

6.3.3.3 array() [3/4]

```
template<typename T >
coder::array< T, 1 >::array (
    const Base & _other ) [inline]

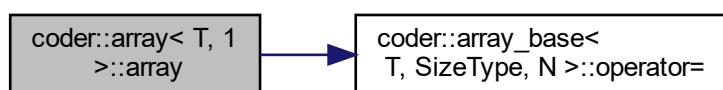
Definition at line 1087 of file coder_array.h.
1088     : Base(_other) {
1089 }
```

6.3.3.4 array() [4/4]

```
template<typename T >
coder::array< T, 1 >::array (
    const std::vector< T > & _vec ) [inline]

Definition at line 1090 of file coder_array.h.
1090 {
1091     operator=(_vec);
1092 }
```

Here is the call graph for this function:



6.3.4 Member Function Documentation

6.3.4.1 at() [1/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1 ) [inline], [inherited]
```

Definition at line 847 of file coder_array.h.

```
847     {
848         ::coder::detail::match_dimensions<N == 1>::check();
849         return data_[_i1];
850     }
```

6.3.4.2 at() [2/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1 ) const [inline], [inherited]
```

Definition at line 888 of file coder_array.h.

```
888     {
889         ::coder::detail::match_dimensions<N == 1>::check();
890         return data_[_i1];
891     }
```

6.3.4.3 at() [3/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) [inline], [inherited]
```

Definition at line 851 of file coder_array.h.

```
851     {
852         ::coder::detail::match_dimensions<N == 2>::check();
853         return data_[index(_i1, _i2)];
854     }
```

6.3.4.4 at() [4/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) const [inline], [inherited]
```

Definition at line 892 of file coder_array.h.

```
892     {
893         ::coder::detail::match_dimensions<N == 2>::check();
894         return data_[index(_i1, _i2)];
895     }
```

6.3.4.5 at() [5/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) [inline], [inherited]
```

Definition at line 855 of file coder_array.h.

```
855             {
856         ::coder::detail::match_dimensions<N == 3>::check();
857         return data_[index(_i1, _i2, _i3)];
858     }
```

6.3.4.6 at() [6/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) const [inline], [inherited]
```

Definition at line 896 of file coder_array.h.

```
896             {
897         ::coder::detail::match_dimensions<N == 3>::check();
898         return data_[index(_i1, _i2, _i3)];
899     }
```

6.3.4.7 at() [7/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) [inline], [inherited]
```

Definition at line 859 of file coder_array.h.

```
859             {
860         ::coder::detail::match_dimensions<N == 4>::check();
861         return data_[index(_i1, _i2, _i3, _i4)];
862     }
```

6.3.4.8 at() [8/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) const [inline], [inherited]
```

Definition at line 900 of file coder_array.h.

```
900             {
901         ::coder::detail::match_dimensions<N == 4>::check();
902         return data_[index(_i1, _i2, _i3, _i4)];
903     }
```

6.3.4.9 at() [9/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) [inline], [inherited]
```

Definition at line 863 of file coder_array.h.

```
863     ::coder::detail::match_dimensions<N == 5>::check();
864     return data_[index(_i1, _i2, _i3, _i4, _i5)];
865 }
866 }
```

6.3.4.10 at() [10/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) const [inline], [inherited]
```

Definition at line 904 of file coder_array.h.

```
904     ::coder::detail::match_dimensions<N == 5>::check();
905     return data_[index(_i1, _i2, _i3, _i4, _i5)];
906 }
907 }
```

6.3.4.11 at() [11/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) [inline], [inherited]
```

Definition at line 867 of file coder_array.h.

```
867     ::coder::detail::match_dimensions<N == 6>::check();
868     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
869 }
870 }
```

6.3.4.12 at() [12/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) const [inline], [inherited]
```

Definition at line 908 of file coder_array.h.

```
908     ::coder::detail::match_dimensions<N == 6>::check();
909     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
910 }
911 }
```

6.3.4.13 at() [13/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) [inline], [inherited]
```

Definition at line 871 of file coder_array.h.

```
871     ::coder::detail::match_dimensions<N == 7>::check();
872     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
873 }
874 }
```

6.3.4.14 at() [14/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) const [inline], [inherited]
```

Definition at line 912 of file coder_array.h.

```
912     ::coder::detail::match_dimensions<N == 7>::check();
913     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
914 }
915 }
```

6.3.4.15 at() [15/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) [inline], [inherited]
```

Definition at line 875 of file coder_array.h.

```
875
876     ::coder::detail::match_dimensions<N == 8>::check();
877     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
878 }
```

6.3.4.16 at() [16/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) const [inline], [inherited]
```

Definition at line 916 of file coder_array.h.

```
916
917     ::coder::detail::match_dimensions<N == 8>::check();
918     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
919 }
```

6.3.4.17 at() [17/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) [inline], [inherited]
```

Definition at line 879 of file coder_array.h.

```
879
880     ::coder::detail::match_dimensions<N == 9>::check();
881     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
882 }
```

6.3.4.18 at() [18/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) const [inline], [inherited]
```

Definition at line 920 of file coder_array.h.

```
920
921     ::coder::detail::match_dimensions<N == 9>::check();
922     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
923 }
```

{

6.3.4.19 at() [19/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) [inline], [inherited]
```

Definition at line 883 of file coder_array.h.

```
883
884     ::coder::detail::match_dimensions<N == 10>::check();
885     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
886 }
```

{

6.3.4.20 at() [20/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) const [inline], [inherited]
```

Definition at line 924 of file coder_array.h.

```
925     {
926     ::coder::detail::match_dimensions<N == 10>::check();
927     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
928 }
```

6.3.4.21 begin() [1/2]

```
array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::begin  
[inline], [inherited]
```

Definition at line 930 of file coder_array.h.

```
930 {  
931     return array_iterator<array_base<T, SZ, N> >(this, 0);  
932 }
```

6.3.4.22 begin() [2/2]

```
const_array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::begin  
[inline], [inherited]
```

Definition at line 936 of file coder_array.h.

```
936 {  
937     return const_array_iterator<array_base<T, SZ, N> >(this, 0);  
938 }
```

6.3.4.23 capacity()

```
SizeType coder::array_base< T, SizeType , N >::capacity [inline], [inherited]
```

Definition at line 595 of file coder_array.h.

```
595 {  
596     return data_.capacity();  
597 }
```

6.3.4.24 clear()

```
void coder::array_base< T, SizeType , N >::clear [inline], [inherited]
```

Definition at line 771 of file coder_array.h.

```
771 {  
772     data_.clear();  
773 }
```

6.3.4.25 data() [1/2]

```
T* coder::array_base< T, SizeType , N >::data [inline], [inherited]
```

Definition at line 775 of file coder_array.h.

```
775 {  
776     return data_;  
777 }
```

6.3.4.26 data() [2/2]

```
const T* coder::array_base< T, SizeType , N >::data [inline], [inherited]
```

Definition at line 779 of file coder_array.h.

```
779         {
780             return data_;
781         }
```

6.3.4.27 end() [1/2]

```
array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::end [inline], [inherited]
```

Definition at line 933 of file coder_array.h.

```
933         {
934             return array_iterator<array_base<T, SZ, N> >(this, this->numel());
935         }
```

6.3.4.28 end() [2/2]

```
const_array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::end [inline], [inherited]
```

Definition at line 939 of file coder_array.h.

```
939         {
940             return const_array_iterator<array_base<T, SZ, N> >(this, this->numel());
941         }
```

6.3.4.29 ensureCapacity()

```
void coder::array_base< T, SizeType , N >::ensureCapacity (
    SizeType _newNumel ) [inline], [private], [inherited]
```

Definition at line 948 of file coder_array.h.

```
948         {
949             if (_newNumel > data_.capacity()) {
950                 SZ i = data_.capacity();
951                 if (i < 16) {
952                     i = 16;
953                 }
954                 while (i < _newNumel) {
955                     if (i > 1073741823) {
956                         i = MAX_int32_T;
957                     } else {
958                         i <= 1;
959                     }
960                 }
961                 data_.reserve(i);
962             }
963             data_.resize(_newNumel);
964         }
965     }
```

6.3.4.30 index() [1/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 795 of file coder_array.h.

```
795      {
796          ::coder::detail::match_dimensions<N == 1>::check ();
797          const SZ indices[] = {_n1};
798          return ::coder::detail::index_nd<1>::compute(size_, indices);
799      }
```

6.3.4.31 index() [2/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 800 of file coder_array.h.

```
800      {
801          ::coder::detail::match_dimensions<N == 2>::check ();
802          const SZ indices[] = {_n1, _n2};
803          return ::coder::detail::index_nd<2>::compute(size_, indices);
804      }
```

6.3.4.32 index() [3/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 805 of file coder_array.h.

```
805      {
806          ::coder::detail::match_dimensions<N == 3>::check ();
807          const SZ indices[] = {_n1, _n2, _n3};
808          return ::coder::detail::index_nd<3>::compute(size_, indices);
809      }
```

6.3.4.33 index() [4/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 810 of file coder_array.h.

```
810      {
811          ::coder::detail::match_dimensions<N == 4>::check ();
812          const SZ indices[] = {_n1, _n2, _n3, _n4};
813          return ::coder::detail::index_nd<4>::compute(size_, indices);
814      }
```

6.3.4.34 index() [5/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 815 of file coder_array.h.

```
815
816     ::coder::detail::match_dimensions<N == 5>::check();
817     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5};
818     return ::coder::detail::index_nd<5>::compute(size_, indices);
819 }
```

6.3.4.35 index() [6/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 820 of file coder_array.h.

```
820
821     ::coder::detail::match_dimensions<N == 6>::check();
822     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6};
823     return ::coder::detail::index_nd<6>::compute(size_, indices);
824 }
```

6.3.4.36 index() [7/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 825 of file coder_array.h.

```
825
826     ::coder::detail::match_dimensions<N == 7>::check();
827     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
828     return ::coder::detail::index_nd<7>::compute(size_, indices);
829 }
```

6.3.4.37 index() [8/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 830 of file coder_array.h.

```
830
831     ::coder::detail::match_dimensions<N == 8>::check();
832     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
833     return ::coder::detail::index_nd<8>::compute(size_, indices);
834 }
```

6.3.4.38 index() [9/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 835 of file coder_array.h.

```
835
836     ::coder::detail::match_dimensions<N == 9>::check();
837     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
838     return ::coder::detail::index_nd<9>::compute(size_, indices);
839 }
```

6.3.4.39 index() [10/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 840 of file coder_array.h.

```
841     {
842         ::coder::detail::match_dimensions<N == 10>::check();
843         const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
844         return ::coder::detail::index_nd<10>::compute(size_, indices);
845     }
```

6.3.4.40 is_owner()

```
bool coder::array_base< T, SizeType , N >::is_owner [inline], [inherited]
```

Definition at line 587 of file coder_array.h.

```
587     {
588         return data_.is_owner();
589     }
```

6.3.4.41 numel()

```
SizeType coder::array_base< T, SizeType , N >::numel [inline], [inherited]
```

Definition at line 791 of file coder_array.h.

```
791     {
792         return ::coder::detail::numel<N>::compute(size_);
793     }
```

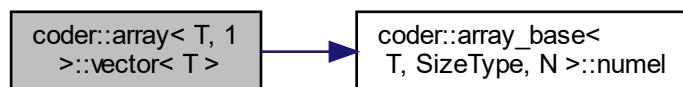
6.3.4.42 operator std::vector< T >()

```
template<typename T >
coder::array< T, 1 >::operator std::vector< T > ( ) const [inline]
```

Definition at line 1101 of file coder_array.h.

```
1101     {
1102         const T* p = &Base::data_[0];
1103         return std::vector<T>(p, p + Base::numel());
1104     }
```

Here is the call graph for this function:



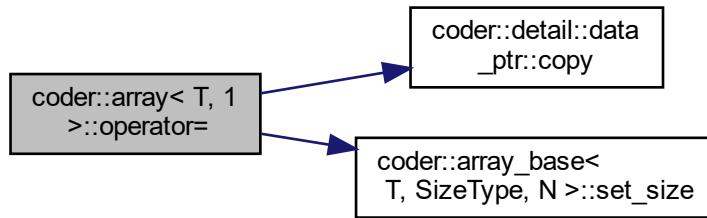
6.3.4.43 operator=()

```
template<typename T >
array& coder::array< T, 1 >::operator= (
    const std::vector< T > & _vec ) [inline]
```

Definition at line 1094 of file coder_array.h.

```
1094 {  
1095     SizeType n = static_cast<SizeType>(_vec.size());  
1096     Base::set_size(n);  
1097     Base::data_.copy(&_vec[0], n);  
1098     return *this;  
1099 }
```

Here is the call graph for this function:



6.3.4.44 operator[]() [1/2]

```
T& coder::array_base< T, SizeType , N >::operator[] (
    SizeType _index ) [inline], [inherited]
```

Definition at line 763 of file coder_array.h.

```
763 {  
764     return data_[_index];  
765 }
```

6.3.4.45 operator[]() [2/2]

```
const T& coder::array_base< T, SizeType , N >::operator[] (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 767 of file coder_array.h.

```
767 {  
768     return data_[_index];  
769 }
```

6.3.4.46 `reshape()` [1/10]

```
array_base<T, SizeType , 1> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 710 of file `coder_array.h`.

```
710
711     const SZ ns[] = {_n1};
712     return reshape_n(ns);
713 }
```

6.3.4.47 `reshape()` [2/10]

```
array_base<T, SizeType , 2> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 715 of file `coder_array.h`.

```
715
716     const SZ ns[] = {_n1, _n2};
717     return reshape_n(ns);
718 }
```

6.3.4.48 `reshape()` [3/10]

```
array_base<T, SizeType , 3> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 720 of file `coder_array.h`.

```
720
721     const SZ ns[] = {_n1, _n2, _n3};
722     return reshape_n(ns);
723 }
```

6.3.4.49 `reshape()` [4/10]

```
array_base<T, SizeType , 4> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 725 of file `coder_array.h`.

```
725
726     const SZ ns[] = {_n1, _n2, _n3, _n4};
727     return reshape_n(ns);
728 }
```

6.3.4.50 reshape() [5/10]

```
array_base<T, SizeType , 5> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 730 of file coder_array.h.

```
730     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5};
731     return reshape_n(ns);
732 }
733 }
```

6.3.4.51 reshape() [6/10]

```
array_base<T, SizeType , 6> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 735 of file coder_array.h.

```
735
736     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6};
737     return reshape_n(ns);
738 }
```

6.3.4.52 reshape() [7/10]

```
array_base<T, SizeType , 7> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 740 of file coder_array.h.

```
740
741     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
742     return reshape_n(ns);
743 }
```

6.3.4.53 reshape() [8/10]

```
array_base<T, SizeType, 8> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 745 of file coder_array.h.

```
746     {
747         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
748         return reshape_n(ns);
749     }
```

6.3.4.54 reshape() [9/10]

```
array_base<T, SizeType, 9> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 752 of file coder_array.h.

```
752     {
753         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
754         return reshape_n(ns);
755     }
```

6.3.4.55 reshape() [10/10]

```
array_base<T, SizeType, 10> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 758 of file coder_array.h.

```
758     {
759         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
760         return reshape_n(ns);
761     }
```

6.3.4.56 reshape_n()

```
array_base<T, SizeType, N1> coder::array_base< T, SizeType , N >::reshape_n (
    const SizeType (&) _ns[N1] ) const [inline], [inherited]
```

Definition at line 705 of file coder_array.h.

```
705     {
706         array_base<T, SZ, N1> reshaped(const_cast<T*>(&data_[0]), _ns);
707         return reshaped;
708     }
```

6.3.4.57 set() [1/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1 ) [inline], [inherited]
```

Definition at line 481 of file coder_array.h.

```
481     {
482         ::coder::detail::match_dimensions<N == 1>::check();
483         data_.set(_data, _n1);
484         size_[0] = _n1;
485     }
```

6.3.4.58 set() [2/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 487 of file coder_array.h.

```
487     {
488         ::coder::detail::match_dimensions<N == 2>::check();
489         data_.set(_data, _n1 * _n2);
490         size_[0] = _n1;
491         size_[1] = _n2;
492     }
```

6.3.4.59 set() [3/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 494 of file coder_array.h.

```
494     {
495         ::coder::detail::match_dimensions<N == 3>::check();
496         data_.set(_data, _n1 * _n2 * _n3);
497         size_[0] = _n1;
498         size_[1] = _n2;
499         size_[2] = _n3;
500     }
```

6.3.4.60 set() [4/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 502 of file coder_array.h.

```
502                                         {
503     ::coder::detail::match_dimensions<N == 4>::check();
504     data_.set(_data, _n1 * _n2 * _n3 * _n4);
505     size_[0] = _n1;
506     size_[1] = _n2;
507     size_[2] = _n3;
508     size_[3] = _n4;
509 }
```

6.3.4.61 set() [5/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 511 of file coder_array.h.

```
511                                         {
512     ::coder::detail::match_dimensions<N == 5>::check();
513     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5);
514     size_[0] = _n1;
515     size_[1] = _n2;
516     size_[2] = _n3;
517     size_[3] = _n4;
518     size_[4] = _n5;
519 }
```

6.3.4.62 set() [6/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 521 of file coder_array.h.

```
521                                         {
522     ::coder::detail::match_dimensions<N == 6>::check();
523     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6);
524     size_[0] = _n1;
525     size_[1] = _n2;
526     size_[2] = _n3;
527     size_[3] = _n4;
528     size_[4] = _n5;
529     size_[5] = _n6;
530 }
```

6.3.4.63 set() [7/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 532 of file coder_array.h.

```
532
533     ::coder::detail::match_dimensions<N == 7>::check();
534     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7);
535     size_[0] = _n1;
536     size_[1] = _n2;
537     size_[2] = _n3;
538     size_[3] = _n4;
539     size_[4] = _n5;
540     size_[5] = _n6;
541     size_[6] = _n7;
542 }
```

6.3.4.64 set() [8/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 544 of file coder_array.h.

```
544
545     ::coder::detail::match_dimensions<N == 8>::check();
546     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8);
547     size_[0] = _n1;
548     size_[1] = _n2;
549     size_[2] = _n3;
550     size_[3] = _n4;
551     size_[4] = _n5;
552     size_[5] = _n6;
553     size_[6] = _n7;
554     size_[7] = _n8;
555 }
```

6.3.4.65 set() [9/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
```

```

    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]

```

Definition at line 557 of file coder_array.h.

```

557
558     ::coder::detail::match_dimensions<N == 9>::check();
559     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9);
560     size_[0] = _n1;
561     size_[1] = _n2;
562     size_[2] = _n3;
563     size_[3] = _n4;
564     size_[4] = _n5;
565     size_[5] = _n6;
566     size_[6] = _n7;
567     size_[7] = _n8;
568     size_[8] = _n9;
569 }

```

6.3.4.66 set() [10/10]

```

void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]

```

Definition at line 572 of file coder_array.h.

```

572
573     ::coder::detail::match_dimensions<N == 10>::check();
574     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9 * _n10);
575     size_[0] = _n1;
576     size_[1] = _n2;
577     size_[2] = _n3;
578     size_[3] = _n4;
579     size_[4] = _n5;
580     size_[5] = _n6;
581     size_[6] = _n7;
582     size_[7] = _n8;
583     size_[8] = _n9;
584     size_[9] = _n10;
585 }

```

6.3.4.67 set_owner()

```

void coder::array_base< T, SizeType , N >::set_owner (
    bool b ) [inline], [inherited]

```

Definition at line 591 of file coder_array.h.

```

591
592     data_.set_owner(b);
593 }

```

6.3.4.68 set_size() [1/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1 ) [inline], [inherited]
```

Definition at line 599 of file coder_array.h.

```
599     {
600         ::coder::detail::match_dimensions<N == 1>::check ();
601         size_[0] = _n1;
602         ensureCapacity(numel ());
603     }
```

6.3.4.69 set_size() [2/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 605 of file coder_array.h.

```
605     {
606         ::coder::detail::match_dimensions<N == 2>::check ();
607         size_[0] = _n1;
608         size_[1] = _n2;
609         ensureCapacity(numel ());
610     }
```

6.3.4.70 set_size() [3/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 612 of file coder_array.h.

```
612     {
613         ::coder::detail::match_dimensions<N == 3>::check ();
614         size_[0] = _n1;
615         size_[1] = _n2;
616         size_[2] = _n3;
617         ensureCapacity(numel ());
618     }
```

6.3.4.71 set_size() [4/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 620 of file coder_array.h.

```
620     {
621         ::coder::detail::match_dimensions<N == 4>::check ();
622         size_[0] = _n1;
623         size_[1] = _n2;
624         size_[2] = _n3;
625         size_[3] = _n4;
626         ensureCapacity(numel ());
627     }
```

6.3.4.72 set_size() [5/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 629 of file coder_array.h.

```
629
630     ::coder::detail::match_dimensions<N == 5>::check();
631     size_[0] = _n1;
632     size_[1] = _n2;
633     size_[2] = _n3;
634     size_[3] = _n4;
635     size_[4] = _n5;
636     ensureCapacity(numel());
637 }
```

6.3.4.73 set_size() [6/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 639 of file coder_array.h.

```
639
640     ::coder::detail::match_dimensions<N == 6>::check();
641     size_[0] = _n1;
642     size_[1] = _n2;
643     size_[2] = _n3;
644     size_[3] = _n4;
645     size_[4] = _n5;
646     size_[5] = _n6;
647     ensureCapacity(numel());
648 }
```

6.3.4.74 set_size() [7/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 650 of file coder_array.h.

```
650
651     ::coder::detail::match_dimensions<N == 7>::check();
652     size_[0] = _n1;
653     size_[1] = _n2;
654     size_[2] = _n3;
655     size_[3] = _n4;
656     size_[4] = _n5;
657     size_[5] = _n6;
658     size_[6] = _n7;
659     ensureCapacity(numel());
660 }
```

6.3.4.75 set_size() [8/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 662 of file coder_array.h.

```
662
663     ::coder::detail::match_dimensions<N == 8>::check();
664     size_[0] = _n1;
665     size_[1] = _n2;
666     size_[2] = _n3;
667     size_[3] = _n4;
668     size_[4] = _n5;
669     size_[5] = _n6;
670     size_[6] = _n7;
671     size_[7] = _n8;
672     ensureCapacity(numel());
673 }
```

6.3.4.76 set_size() [9/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]
```

Definition at line 675 of file coder_array.h.

```
675
676     ::coder::detail::match_dimensions<N == 9>::check();
677     size_[0] = _n1;
678     size_[1] = _n2;
679     size_[2] = _n3;
680     size_[3] = _n4;
681     size_[4] = _n5;
682     size_[5] = _n6;
683     size_[6] = _n7;
684     size_[7] = _n8;
685     size_[8] = _n9;
686     ensureCapacity(numel());
687 }
```

6.3.4.77 set_size() [10/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]
```

Definition at line 689 of file coder_array.h.

```
689
690     ::coder::detail::match_dimensions<N == 10>::check();
691     size_[0] = _n1;
692     size_[1] = _n2;
693     size_[2] = _n3;
694     size_[3] = _n4;
695     size_[4] = _n5;
696     size_[5] = _n6;
697     size_[6] = _n7;
698     size_[7] = _n8;
699     size_[8] = _n9;
700     size_[9] = _n10;
701     ensureCapacity(numel());
702 }
```

6.3.4.78 size() [1/2]

```
const SizeType * coder::array_base< T, SizeType , N >::size [inline], [inherited]
```

Definition at line 783 of file coder_array.h.

```
783
784     return &size_[0];
785 }
```

6.3.4.79 size() [2/2]

```
SizeType coder::array_base< T, SizeType , N >::size (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 787 of file coder_array.h.

```
787
788     return size_[_index];
789 }
```

6.3.5 Member Data Documentation

6.3.5.1 data_

```
::coder::detail::data_ptr<T, SizeType> coder::array_base< T, SizeType , N >::data_ [protected],  
[inherited]
```

Definition at line 944 of file [coder_array.h](#).

6.3.5.2 size_

```
SizeType coder::array_base< T, SizeType , N >::size_[N] [protected], [inherited]
```

Definition at line 945 of file [coder_array.h](#).

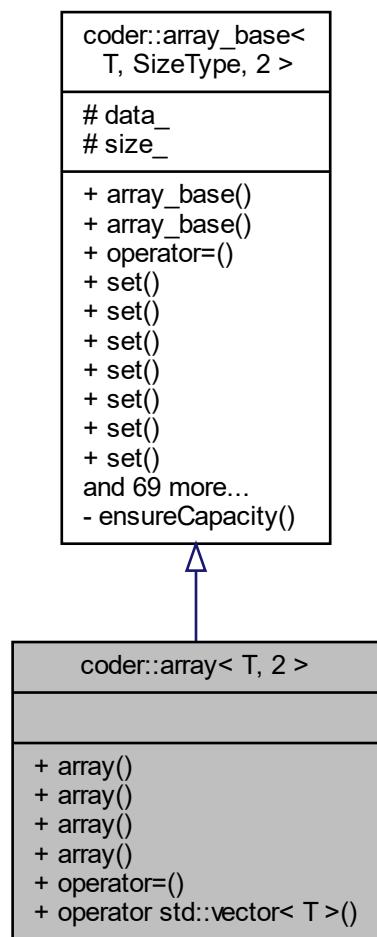
The documentation for this class was generated from the following file:

- codegen/lib/Func_fovPathPlanning/[coder_array.h](#)

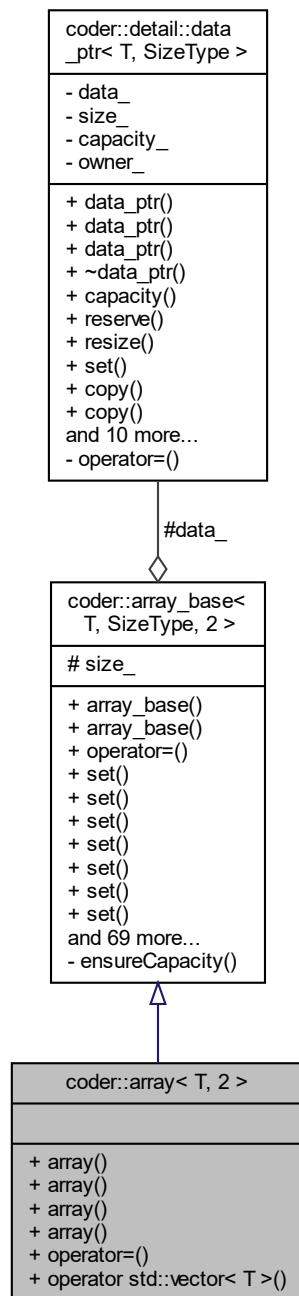
6.4 coder::array< T, 2 > Class Template Reference

```
#include <coder_array.h>
```

Inheritance diagram for coder::array< T, 2 >:



Collaboration diagram for coder::array< T, 2 >:



Public Types

- `typedef T value_type`
- `typedef SizeType size_type`

Public Member Functions

- `array ()`

- `array (const array< T, 2 > &_other)`
- `array (const Base &_other)`
- `array (const std::vector< T > &_vec)`
- `array & operator= (const std::vector< T > &_vec)`
- `operator std::vector< T > () const`
- `void set (T *_data, SizeType _n1)`
- `void set (T *_data, SizeType _n1, SizeType _n2)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set (T *_data, SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `bool is_owner () const`
- `void set_owner (bool b)`
- `SizeType capacity () const`
- `void set_size (SizeType _n1)`
- `void set_size (SizeType _n1, SizeType _n2)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9)`
- `void set_size (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10)`
- `array_base< T, SizeType, N1 > reshape_n (const SizeType(&_ns)[N1]) const`
- `array_base< T, SizeType, 1 > reshape (SizeType _n1) const`
- `array_base< T, SizeType, 2 > reshape (SizeType _n1, SizeType _n2) const`
- `array_base< T, SizeType, 3 > reshape (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `array_base< T, SizeType, 4 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `array_base< T, SizeType, 5 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `array_base< T, SizeType, 6 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `array_base< T, SizeType, 7 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `array_base< T, SizeType, 8 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `array_base< T, SizeType, 9 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`
- `array_base< T, SizeType, 10 > reshape (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- `T & operator[] (SizeType _index)`
- `const T & operator[] (SizeType _index) const`

- void `clear ()`
- `T * data ()`
- `const T * data () const`
- `const SizeType * size () const`
- `SizeType size (SizeType _index) const`
- `SizeType numel () const`
- `SizeType index (SizeType _n1) const`
- `SizeType index (SizeType _n1, SizeType _n2) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9) const`
- `SizeType index (SizeType _n1, SizeType _n2, SizeType _n3, SizeType _n4, SizeType _n5, SizeType _n6, SizeType _n7, SizeType _n8, SizeType _n9, SizeType _n10) const`
- `T & at (SizeType _i1)`
- `T & at (SizeType _i1, SizeType _i2)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9)`
- `T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10)`
- `const T & at (SizeType _i1) const`
- `const T & at (SizeType _i1, SizeType _i2) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9) const`
- `const T & at (SizeType _i1, SizeType _i2, SizeType _i3, SizeType _i4, SizeType _i5, SizeType _i6, SizeType _i7, SizeType _i8, SizeType _i9, SizeType _i10) const`
- `array_iterator< array_base< T, SizeType, N > > begin ()`
- `const_array_iterator< array_base< T, SizeType, N > > begin () const`
- `array_iterator< array_base< T, SizeType, N > > end ()`
- `const_array_iterator< array_base< T, SizeType, N > > end () const`

Protected Attributes

- `::coder::detail::data_ptr< T, SizeType > data_`
- `SizeType size_[N]`

Private Types

- `typedef array_base< T, SizeType, 2 > Base`

Private Member Functions

- `void ensureCapacity (SizeType _newNumel)`

6.4.1 Detailed Description

```
template<typename T>
class coder::array< T, 2 >
```

Definition at line 1042 of file coder_array.h.

6.4.2 Member Typedef Documentation

6.4.2.1 Base

```
template<typename T >
typedef array_base<T, SizeType, 2> coder::array< T, 2 >::Base [private]
```

Definition at line 1044 of file coder_array.h.

6.4.2.2 size_type

```
typedef SizeType coder::array_base< T, SizeType , N >::size_type [inherited]
```

Definition at line 464 of file coder_array.h.

6.4.2.3 value_type

```
typedef T coder::array_base< T, SizeType , N >::value_type [inherited]
```

Definition at line 463 of file coder_array.h.

6.4.3 Constructor & Destructor Documentation

6.4.3.1 array() [1/4]

```
template<typename T >
coder::array< T, 2 >::array ( ) [inline]

Definition at line 1047 of file coder_array.h.
1048     : Base() {
1049 }
```

6.4.3.2 array() [2/4]

```
template<typename T >
coder::array< T, 2 >::array (
    const array< T, 2 > & _other ) [inline]

Definition at line 1050 of file coder_array.h.
1051     : Base(_other) {
1052 }
```

6.4.3.3 array() [3/4]

```
template<typename T >
coder::array< T, 2 >::array (
    const Base & _other ) [inline]

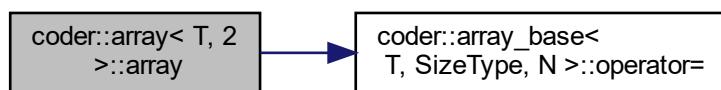
Definition at line 1053 of file coder_array.h.
1054     : Base(_other) {
1055 }
```

6.4.3.4 array() [4/4]

```
template<typename T >
coder::array< T, 2 >::array (
    const std::vector< T > & _vec ) [inline]

Definition at line 1056 of file coder_array.h.
1056 {
1057     operator=(_vec);
1058 }
```

Here is the call graph for this function:



6.4.4 Member Function Documentation

6.4.4.1 at() [1/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1 ) [inline], [inherited]
```

Definition at line 847 of file coder_array.h.

```
847     {
848         ::coder::detail::match_dimensions<N == 1>::check();
849         return data_[_i1];
850     }
```

6.4.4.2 at() [2/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1 ) const [inline], [inherited]
```

Definition at line 888 of file coder_array.h.

```
888     {
889         ::coder::detail::match_dimensions<N == 1>::check();
890         return data_[_i1];
891     }
```

6.4.4.3 at() [3/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) [inline], [inherited]
```

Definition at line 851 of file coder_array.h.

```
851     {
852         ::coder::detail::match_dimensions<N == 2>::check();
853         return data_[index(_i1, _i2)];
854     }
```

6.4.4.4 at() [4/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2 ) const [inline], [inherited]
```

Definition at line 892 of file coder_array.h.

```
892     {
893         ::coder::detail::match_dimensions<N == 2>::check();
894         return data_[index(_i1, _i2)];
895     }
```

6.4.4.5 at() [5/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) [inline], [inherited]
```

Definition at line 855 of file coder_array.h.

```
855             {
856         ::coder::detail::match_dimensions<N == 3>::check();
857         return data_[index(_i1, _i2, _i3)];
858     }
```

6.4.4.6 at() [6/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3 ) const [inline], [inherited]
```

Definition at line 896 of file coder_array.h.

```
896             {
897         ::coder::detail::match_dimensions<N == 3>::check();
898         return data_[index(_i1, _i2, _i3)];
899     }
```

6.4.4.7 at() [7/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) [inline], [inherited]
```

Definition at line 859 of file coder_array.h.

```
859             {
860         ::coder::detail::match_dimensions<N == 4>::check();
861         return data_[index(_i1, _i2, _i3, _i4)];
862     }
```

6.4.4.8 at() [8/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4 ) const [inline], [inherited]
```

Definition at line 900 of file coder_array.h.

```
900             {
901         ::coder::detail::match_dimensions<N == 4>::check();
902         return data_[index(_i1, _i2, _i3, _i4)];
903     }
```

6.4.4.9 at() [9/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) [inline], [inherited]
```

Definition at line 863 of file coder_array.h.

```
863
864     ::coder::detail::match_dimensions<N == 5>::check();
865     return data_[index(_i1, _i2, _i3, _i4, _i5)];
866 }
```

6.4.4.10 at() [10/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5 ) const [inline], [inherited]
```

Definition at line 904 of file coder_array.h.

```
904
905     ::coder::detail::match_dimensions<N == 5>::check();
906     return data_[index(_i1, _i2, _i3, _i4, _i5)];
907 }
```

6.4.4.11 at() [11/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) [inline], [inherited]
```

Definition at line 867 of file coder_array.h.

```
867
868     ::coder::detail::match_dimensions<N == 6>::check();
869     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
870 }
```

6.4.4.12 at() [12/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6 ) const [inline], [inherited]
```

Definition at line 908 of file coder_array.h.

```
908     ::coder::detail::match_dimensions<N == 6>::check();
909     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
910 }
911 }
```

6.4.4.13 at() [13/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) [inline], [inherited]
```

Definition at line 871 of file coder_array.h.

```
871     ::coder::detail::match_dimensions<N == 7>::check();
872     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
873 }
874 }
```

6.4.4.14 at() [14/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7 ) const [inline], [inherited]
```

Definition at line 912 of file coder_array.h.

```
912     ::coder::detail::match_dimensions<N == 7>::check();
913     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
914 }
915 }
```

6.4.4.15 at() [15/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) [inline], [inherited]
```

Definition at line 875 of file coder_array.h.

```
875
876     ::coder::detail::match_dimensions<N == 8>::check();
877     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
878 }
```

6.4.4.16 at() [16/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8 ) const [inline], [inherited]
```

Definition at line 916 of file coder_array.h.

```
916
917     ::coder::detail::match_dimensions<N == 8>::check();
918     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
919 }
```

6.4.4.17 at() [17/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) [inline], [inherited]
```

Definition at line 879 of file coder_array.h.

```
879
880     ::coder::detail::match_dimensions<N == 9>::check();
881     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
882 }
```

6.4.4.18 at() [18/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9 ) const [inline], [inherited]
```

Definition at line 920 of file coder_array.h.

```
920
921     ::coder::detail::match_dimensions<N == 9>::check();
922     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
923 }
```

6.4.4.19 at() [19/20]

```
T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) [inline], [inherited]
```

Definition at line 883 of file coder_array.h.

```
883
884     ::coder::detail::match_dimensions<N == 10>::check();
885     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
886 }
```

6.4.4.20 at() [20/20]

```
const T& coder::array_base< T, SizeType , N >::at (
    SizeType _i1,
    SizeType _i2,
    SizeType _i3,
    SizeType _i4,
    SizeType _i5,
    SizeType _i6,
    SizeType _i7,
    SizeType _i8,
    SizeType _i9,
    SizeType _i10 ) const [inline], [inherited]
```

Definition at line 924 of file coder_array.h.

```
925     {
926     ::coder::detail::match_dimensions<N == 10>::check();
927     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
928 }
```

6.4.4.21 `begin()` [1/2]

```
array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::begin
[inline], [inherited]
```

Definition at line 930 of file `coder_array.h`.

```
930
931     return array_iterator<array_base<T, SZ, N> >(this, 0);
932 }
```

6.4.4.22 `begin()` [2/2]

```
const_array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::begin
[inline], [inherited]
```

Definition at line 936 of file `coder_array.h`.

```
936
937     return const_array_iterator<array_base<T, SZ, N> >(this, 0);
938 }
```

6.4.4.23 `capacity()`

```
SizeType coder::array_base< T, SizeType , N >::capacity [inline], [inherited]
```

Definition at line 595 of file `coder_array.h`.

```
595
596     return data_.capacity();
597 }
```

6.4.4.24 `clear()`

```
void coder::array_base< T, SizeType , N >::clear [inline], [inherited]
```

Definition at line 771 of file `coder_array.h`.

```
771
772     data_.clear();
773 }
```

6.4.4.25 `data()` [1/2]

```
T* coder::array_base< T, SizeType , N >::data [inline], [inherited]
```

Definition at line 775 of file `coder_array.h`.

```
775
776     return data_;
777 }
```

6.4.4.26 data() [2/2]

```
const T* coder::array_base< T, SizeType , N >::data [inline], [inherited]
```

Definition at line 779 of file coder_array.h.

```
779         {
780             return data_;
781         }
```

6.4.4.27 end() [1/2]

```
array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::end [inline], [inherited]
```

Definition at line 933 of file coder_array.h.

```
933         {
934             return array_iterator<array_base<T, SZ, N> >(this, this->numel());
935         }
```

6.4.4.28 end() [2/2]

```
const_array_iterator<array_base<T, SizeType , N> > coder::array_base< T, SizeType , N >::end [inline], [inherited]
```

Definition at line 939 of file coder_array.h.

```
939         {
940             return const_array_iterator<array_base<T, SZ, N> >(this, this->numel());
941         }
```

6.4.4.29 ensureCapacity()

```
void coder::array_base< T, SizeType , N >::ensureCapacity (
    SizeType _newNumel ) [inline], [private], [inherited]
```

Definition at line 948 of file coder_array.h.

```
948         {
949             if (_newNumel > data_.capacity()) {
950                 SZ i = data_.capacity();
951                 if (i < 16) {
952                     i = 16;
953                 }
954                 while (i < _newNumel) {
955                     if (i > 1073741823) {
956                         i = MAX_int32_T;
957                     } else {
958                         i <<= 1;
959                     }
960                 }
961                 data_.reserve(i);
962             }
963             data_.resize(_newNumel);
964         }
965     }
```

6.4.4.30 index() [1/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 795 of file coder_array.h.

```
795      {
796          ::coder::detail::match_dimensions<N == 1>::check ();
797          const SZ indices[] = {_n1};
798          return ::coder::detail::index_nd<1>::compute(size_, indices);
799      }
```

6.4.4.31 index() [2/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 800 of file coder_array.h.

```
800      {
801          ::coder::detail::match_dimensions<N == 2>::check ();
802          const SZ indices[] = {_n1, _n2};
803          return ::coder::detail::index_nd<2>::compute(size_, indices);
804      }
```

6.4.4.32 index() [3/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 805 of file coder_array.h.

```
805      {
806          ::coder::detail::match_dimensions<N == 3>::check ();
807          const SZ indices[] = {_n1, _n2, _n3};
808          return ::coder::detail::index_nd<3>::compute(size_, indices);
809      }
```

6.4.4.33 index() [4/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 810 of file coder_array.h.

```
810      {
811          ::coder::detail::match_dimensions<N == 4>::check ();
812          const SZ indices[] = {_n1, _n2, _n3, _n4};
813          return ::coder::detail::index_nd<4>::compute(size_, indices);
814      }
```

6.4.4.34 index() [5/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 815 of file coder_array.h.

```
815
816     ::coder::detail::match_dimensions<N == 5>::check();
817     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5};
818     return ::coder::detail::index_nd<5>::compute(size_, indices);
819 }
```

6.4.4.35 index() [6/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 820 of file coder_array.h.

```
820
821     ::coder::detail::match_dimensions<N == 6>::check();
822     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6};
823     return ::coder::detail::index_nd<6>::compute(size_, indices);
824 }
```

6.4.4.36 index() [7/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 825 of file coder_array.h.

```
825
826     ::coder::detail::match_dimensions<N == 7>::check();
827     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
828     return ::coder::detail::index_nd<7>::compute(size_, indices);
829 }
```

6.4.4.37 index() [8/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 830 of file coder_array.h.

```
830
831     ::coder::detail::match_dimensions<N == 8>::check();
832     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
833     return ::coder::detail::index_nd<8>::compute(size_, indices);
834 }
```

6.4.4.38 index() [9/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 835 of file coder_array.h.

```
835
836     ::coder::detail::match_dimensions<N == 9>::check();
837     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
838     return ::coder::detail::index_nd<9>::compute(size_, indices);
839 }
```

6.4.4.39 index() [10/10]

```
SizeType coder::array_base< T, SizeType , N >::index (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 840 of file coder_array.h.

```
841     {
842         ::coder::detail::match_dimensions<N == 10>::check();
843         const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
844         return ::coder::detail::index_nd<10>::compute(size_, indices);
845     }
```

6.4.4.40 is_owner()

```
bool coder::array_base< T, SizeType , N >::is_owner [inline], [inherited]
```

Definition at line 587 of file coder_array.h.

```
587         {
588             return data_.is_owner();
589 }
```

6.4.4.41 numel()

```
SizeType coder::array_base< T, SizeType , N >::numel [inline], [inherited]
```

Definition at line 791 of file coder_array.h.

```
791         {
792             return ::coder::detail::numel<N>::compute(size_);
793 }
```

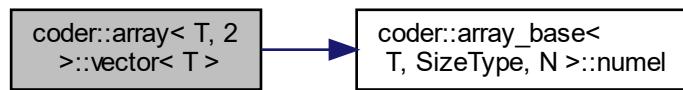
6.4.4.42 operator std::vector< T >()

```
template<typename T >
coder::array< T, 2 >::operator std::vector< T > ( ) const [inline]
```

Definition at line 1067 of file coder_array.h.

```
1067         {
1068             const T* p = &Base::data_[0];
1069             return std::vector<T>(p, p + Base::numel());
1070 }
```

Here is the call graph for this function:



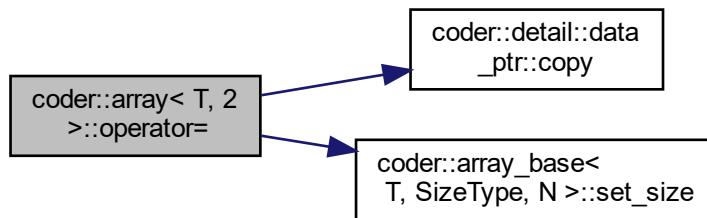
6.4.4.43 operator=()

```
template<typename T >
array& coder::array< T, 2 >::operator= (
    const std::vector< T > & _vec ) [inline]
```

Definition at line 1060 of file coder_array.h.

```
1060                                     {
1061     SizeType n = static_cast<SizeType>(_vec.size());
1062     Base::set_size(1, n);
1063     Base::data_.copy(&_vec[0], n);
1064     return *this;
1065 }
```

Here is the call graph for this function:



6.4.4.44 operator[]() [1/2]

```
T& coder::array_base< T, SizeType , N >::operator[] (
    SizeType _index ) [inline], [inherited]
```

Definition at line 763 of file coder_array.h.

```
763 {
764     return data_[_index];
765 }
```

6.4.4.45 operator[]() [2/2]

```
const T& coder::array_base< T, SizeType , N >::operator[] (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 767 of file coder_array.h.

```
767 {
768     return data_[_index];
769 }
```

6.4.4.46 reshape() [1/10]

```
array_base<T, SizeType , 1> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1 ) const [inline], [inherited]
```

Definition at line 710 of file coder_array.h.

```
710
711     const SZ ns[] = {_n1};
712     return reshape_n(ns);
713 }
```

6.4.4.47 reshape() [2/10]

```
array_base<T, SizeType , 2> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2 ) const [inline], [inherited]
```

Definition at line 715 of file coder_array.h.

```
715
716     const SZ ns[] = {_n1, _n2};
717     return reshape_n(ns);
718 }
```

6.4.4.48 reshape() [3/10]

```
array_base<T, SizeType , 3> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) const [inline], [inherited]
```

Definition at line 720 of file coder_array.h.

```
720
721     const SZ ns[] = {_n1, _n2, _n3};
722     return reshape_n(ns);
723 }
```

6.4.4.49 reshape() [4/10]

```
array_base<T, SizeType , 4> coder::array_base< T, SizeType , N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) const [inline], [inherited]
```

Definition at line 725 of file coder_array.h.

```
725
726     const SZ ns[] = {_n1, _n2, _n3, _n4};
727     return reshape_n(ns);
728 }
```

6.4.4.50 `reshape()` [5/10]

```
array_base<T, SizeType, 5> coder::array_base< T, SizeType, N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) const [inline], [inherited]
```

Definition at line 730 of file `coder_array.h`.

```
730     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5};
731     return reshape_n(ns);
732 }
733 }
```

6.4.4.51 `reshape()` [6/10]

```
array_base<T, SizeType, 6> coder::array_base< T, SizeType, N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) const [inline], [inherited]
```

Definition at line 735 of file `coder_array.h`.

```
735
736     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6};
737     return reshape_n(ns);
738 }
```

6.4.4.52 `reshape()` [7/10]

```
array_base<T, SizeType, 7> coder::array_base< T, SizeType, N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) const [inline], [inherited]
```

Definition at line 740 of file `coder_array.h`.

```
740
741     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
742     return reshape_n(ns);
743 }
```

6.4.4.53 reshape() [8/10]

```
array_base<T, SizeType, 8> coder::array_base< T, SizeType, N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) const [inline], [inherited]
```

Definition at line 745 of file coder_array.h.

```
746     {
747         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
748         return reshape_n(ns);
749     }
```

6.4.4.54 reshape() [9/10]

```
array_base<T, SizeType, 9> coder::array_base< T, SizeType, N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) const [inline], [inherited]
```

Definition at line 752 of file coder_array.h.

```
752     {
753         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
754         return reshape_n(ns);
755     }
```

6.4.4.55 reshape() [10/10]

```
array_base<T, SizeType, 10> coder::array_base< T, SizeType, N >::reshape (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) const [inline], [inherited]
```

Definition at line 758 of file coder_array.h.

```
758     {
759         const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
760         return reshape_n(ns);
761     }
```

6.4.4.56 reshape_n()

```
array_base<T, SizeType, N1> coder::array_base< T, SizeType , N >::reshape_n (
    const SizeType (&) _ns[N1] ) const [inline], [inherited]
```

Definition at line 705 of file coder_array.h.

```
705     {
706         array_base<T, SZ, N1> reshaped(const_cast<T*>(&data_[0]), _ns);
707         return reshaped;
708     }
```

6.4.4.57 set() [1/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1 ) [inline], [inherited]
```

Definition at line 481 of file coder_array.h.

```
481     {
482         ::coder::detail::match_dimensions<N == 1>::check();
483         data_.set(_data, _n1);
484         size_[0] = _n1;
485     }
```

6.4.4.58 set() [2/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 487 of file coder_array.h.

```
487     {
488         ::coder::detail::match_dimensions<N == 2>::check();
489         data_.set(_data, _n1 * _n2);
490         size_[0] = _n1;
491         size_[1] = _n2;
492     }
```

6.4.4.59 set() [3/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 494 of file coder_array.h.

```
494     {
495         ::coder::detail::match_dimensions<N == 3>::check();
496         data_.set(_data, _n1 * _n2 * _n3);
497         size_[0] = _n1;
498         size_[1] = _n2;
499         size_[2] = _n3;
500     }
```

6.4.4.60 set() [4/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 502 of file coder_array.h.

```
502 {
503     ::coder::detail::match_dimensions<N == 4>::check();
504     data_.set(_data, _n1 * _n2 * _n3 * _n4);
505     size_[0] = _n1;
506     size_[1] = _n2;
507     size_[2] = _n3;
508     size_[3] = _n4;
509 }
```

6.4.4.61 set() [5/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 511 of file coder_array.h.

```
511 {
512     ::coder::detail::match_dimensions<N == 5>::check();
513     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5);
514     size_[0] = _n1;
515     size_[1] = _n2;
516     size_[2] = _n3;
517     size_[3] = _n4;
518     size_[4] = _n5;
519 }
```

6.4.4.62 set() [6/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 521 of file coder_array.h.

```
521 {
522     ::coder::detail::match_dimensions<N == 6>::check();
523     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6);
524     size_[0] = _n1;
525     size_[1] = _n2;
526     size_[2] = _n3;
527     size_[3] = _n4;
528     size_[4] = _n5;
529     size_[5] = _n6;
530 }
```

6.4.4.63 set() [7/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 532 of file coder_array.h.

```
532
533     ::coder::detail::match_dimensions<N == 7>::check();
534     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7);
535     size_[0] = _n1;
536     size_[1] = _n2;
537     size_[2] = _n3;
538     size_[3] = _n4;
539     size_[4] = _n5;
540     size_[5] = _n6;
541     size_[6] = _n7;
542 }
```

6.4.4.64 set() [8/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 544 of file coder_array.h.

```
544
545     ::coder::detail::match_dimensions<N == 8>::check();
546     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8);
547     size_[0] = _n1;
548     size_[1] = _n2;
549     size_[2] = _n3;
550     size_[3] = _n4;
551     size_[4] = _n5;
552     size_[5] = _n6;
553     size_[6] = _n7;
554     size_[7] = _n8;
555 }
```

6.4.4.65 set() [9/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
```

```
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]
```

Definition at line 557 of file coder_array.h.

```
557
558     ::coder::detail::match_dimensions<N == 9>::check();
559     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9);
560     size_[0] = _n1;
561     size_[1] = _n2;
562     size_[2] = _n3;
563     size_[3] = _n4;
564     size_[4] = _n5;
565     size_[5] = _n6;
566     size_[6] = _n7;
567     size_[7] = _n8;
568     size_[8] = _n9;
569 }
```

6.4.4.66 set() [10/10]

```
void coder::array_base< T, SizeType , N >::set (
    T * _data,
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]
```

Definition at line 572 of file coder_array.h.

```
572
573     ::coder::detail::match_dimensions<N == 10>::check();
574     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9 * _n10);
575     size_[0] = _n1;
576     size_[1] = _n2;
577     size_[2] = _n3;
578     size_[3] = _n4;
579     size_[4] = _n5;
580     size_[5] = _n6;
581     size_[6] = _n7;
582     size_[7] = _n8;
583     size_[8] = _n9;
584     size_[9] = _n10;
585 }
```

6.4.4.67 set_owner()

```
void coder::array_base< T, SizeType , N >::set_owner (
    bool b ) [inline], [inherited]
```

Definition at line 591 of file coder_array.h.

```
591
592     data_.set_owner(b);
593 }
```

6.4.4.68 set_size() [1/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1 ) [inline], [inherited]
```

Definition at line 599 of file coder_array.h.

```
599     {
600         ::coder::detail::match_dimensions<N == 1>::check ();
601         size_[0] = _n1;
602         ensureCapacity(numel ());
603     }
```

6.4.4.69 set_size() [2/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2 ) [inline], [inherited]
```

Definition at line 605 of file coder_array.h.

```
605     {
606         ::coder::detail::match_dimensions<N == 2>::check ();
607         size_[0] = _n1;
608         size_[1] = _n2;
609         ensureCapacity(numel ());
610     }
```

6.4.4.70 set_size() [3/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3 ) [inline], [inherited]
```

Definition at line 612 of file coder_array.h.

```
612     {
613         ::coder::detail::match_dimensions<N == 3>::check ();
614         size_[0] = _n1;
615         size_[1] = _n2;
616         size_[2] = _n3;
617         ensureCapacity(numel ());
618     }
```

6.4.4.71 set_size() [4/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4 ) [inline], [inherited]
```

Definition at line 620 of file coder_array.h.

```
620     {
621         ::coder::detail::match_dimensions<N == 4>::check ();
622         size_[0] = _n1;
623         size_[1] = _n2;
624         size_[2] = _n3;
625         size_[3] = _n4;
626         ensureCapacity(numel ());
627     }
```

6.4.4.72 set_size() [5/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5 ) [inline], [inherited]
```

Definition at line 629 of file coder_array.h.

```
629
630     ::coder::detail::match_dimensions<N == 5>::check();
631     size_[0] = _n1;
632     size_[1] = _n2;
633     size_[2] = _n3;
634     size_[3] = _n4;
635     size_[4] = _n5;
636     ensureCapacity(numel());
637 }
```

6.4.4.73 set_size() [6/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6 ) [inline], [inherited]
```

Definition at line 639 of file coder_array.h.

```
639
640     ::coder::detail::match_dimensions<N == 6>::check();
641     size_[0] = _n1;
642     size_[1] = _n2;
643     size_[2] = _n3;
644     size_[3] = _n4;
645     size_[4] = _n5;
646     size_[5] = _n6;
647     ensureCapacity(numel());
648 }
```

6.4.4.74 set_size() [7/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7 ) [inline], [inherited]
```

Definition at line 650 of file coder_array.h.

```
650
651     ::coder::detail::match_dimensions<N == 7>::check();
652     size_[0] = _n1;
653     size_[1] = _n2;
654     size_[2] = _n3;
655     size_[3] = _n4;
656     size_[4] = _n5;
657     size_[5] = _n6;
658     size_[6] = _n7;
659     ensureCapacity(numel());
660 }
```

6.4.4.75 `set_size()` [8/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8 ) [inline], [inherited]
```

Definition at line 662 of file `coder_array.h`.

```
662
663     ::coder::detail::match_dimensions<N == 8>::check();
664     size_[0] = _n1;
665     size_[1] = _n2;
666     size_[2] = _n3;
667     size_[3] = _n4;
668     size_[4] = _n5;
669     size_[5] = _n6;
670     size_[6] = _n7;
671     size_[7] = _n8;
672     ensureCapacity(numel());
673 }
```

6.4.4.76 `set_size()` [9/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9 ) [inline], [inherited]
```

Definition at line 675 of file `coder_array.h`.

```
675
676     ::coder::detail::match_dimensions<N == 9>::check();
677     size_[0] = _n1;
678     size_[1] = _n2;
679     size_[2] = _n3;
680     size_[3] = _n4;
681     size_[4] = _n5;
682     size_[5] = _n6;
683     size_[6] = _n7;
684     size_[7] = _n8;
685     size_[8] = _n9;
686     ensureCapacity(numel());
687 }
```

6.4.4.77 set_size() [10/10]

```
void coder::array_base< T, SizeType , N >::set_size (
    SizeType _n1,
    SizeType _n2,
    SizeType _n3,
    SizeType _n4,
    SizeType _n5,
    SizeType _n6,
    SizeType _n7,
    SizeType _n8,
    SizeType _n9,
    SizeType _n10 ) [inline], [inherited]
```

Definition at line 689 of file coder_array.h.

```
689
690     ::coder::detail::match_dimensions<N == 10>::check();
691     size_[0] = _n1;
692     size_[1] = _n2;
693     size_[2] = _n3;
694     size_[3] = _n4;
695     size_[4] = _n5;
696     size_[5] = _n6;
697     size_[6] = _n7;
698     size_[7] = _n8;
699     size_[8] = _n9;
700     size_[9] = _n10;
701     ensureCapacity(numel());
702 }
```

6.4.4.78 size() [1/2]

```
const SizeType * coder::array_base< T, SizeType , N >::size [inline], [inherited]
```

Definition at line 783 of file coder_array.h.

```
783
784     return &size_[0];
785 }
```

6.4.4.79 size() [2/2]

```
SizeType coder::array_base< T, SizeType , N >::size (
    SizeType _index ) const [inline], [inherited]
```

Definition at line 787 of file coder_array.h.

```
787
788     return size_[_index];
789 }
```

6.4.5 Member Data Documentation

6.4.5.1 `data_`

```
::coder::detail::data_ptr<T, SizeType > coder::array_base< T, SizeType , N >::data_ [protected],  
[inherited]
```

Definition at line 944 of file `coder_array.h`.

6.4.5.2 `size_`

```
SizeType coder::array_base< T, SizeType , N >::size_[N] [protected], [inherited]
```

Definition at line 945 of file `coder_array.h`.

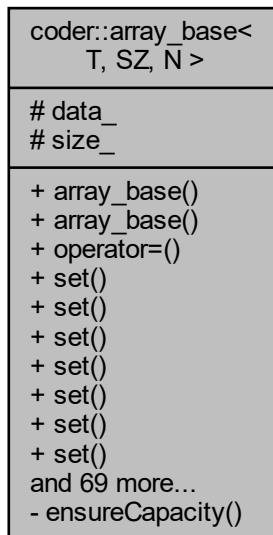
The documentation for this class was generated from the following file:

- `codegen/lib(Func_fovPathPlanning/coder_array.h`

6.5 `coder::array_base< T, SZ, N >` Class Template Reference

```
#include <coder_array.h>
```

Collaboration diagram for `coder::array_base< T, SZ, N >`:



Public Types

- `typedef T value_type`
- `typedef SZ size_type`

Public Member Functions

- `array_base ()`
- `array_base (T *_data, const SZ *_sz)`
- `array_base & operator= (const array_base &_other)`
- `void set (T *_data, SZ _n1)`
- `void set (T *_data, SZ _n1, SZ _n2)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8, SZ _n9)`
- `void set (T *_data, SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8, SZ _n9, SZ _n10)`
- `bool is_owner () const`
- `void set_owner (bool b)`
- `SZ capacity () const`
- `void set_size (SZ _n1)`
- `void set_size (SZ _n1, SZ _n2)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8, SZ _n9)`
- `void set_size (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8, SZ _n9, SZ _n10)`
- template<size_t N1>
 - `array_base< T, SZ, N1 > reshape_n (const SZ(&_ns)[N1]) const`
- `array_base< T, SZ, 1 > reshape (SZ _n1) const`
- `array_base< T, SZ, 2 > reshape (SZ _n1, SZ _n2) const`
- `array_base< T, SZ, 3 > reshape (SZ _n1, SZ _n2, SZ _n3) const`
- `array_base< T, SZ, 4 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4) const`
- `array_base< T, SZ, 5 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5) const`
- `array_base< T, SZ, 6 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6) const`
- `array_base< T, SZ, 7 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7) const`
- `array_base< T, SZ, 8 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8) const`
- `array_base< T, SZ, 9 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8, SZ _n9) const`
- `array_base< T, SZ, 10 > reshape (SZ _n1, SZ _n2, SZ _n3, SZ _n4, SZ _n5, SZ _n6, SZ _n7, SZ _n8, SZ _n9, SZ _n10) const`
- `T & operator[] (SZ _index)`
- `const T & operator[] (SZ _index) const`
- `void clear ()`
- `T * data ()`
- `const T * data () const`
- `const SZ * size () const`
- `SZ size (SZ _index) const`

- `SZ numel () const`
- `SZ index (SZ_n1) const`
- `SZ index (SZ_n1, SZ_n2) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4, SZ_n5) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4, SZ_n5, SZ_n6) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4, SZ_n5, SZ_n6, SZ_n7) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4, SZ_n5, SZ_n6, SZ_n7, SZ_n8) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4, SZ_n5, SZ_n6, SZ_n7, SZ_n8, SZ_n9) const`
- `SZ index (SZ_n1, SZ_n2, SZ_n3, SZ_n4, SZ_n5, SZ_n6, SZ_n7, SZ_n8, SZ_n9, SZ_n10) const`
- `T & at (SZ_i1)`
- `T & at (SZ_i1, SZ_i2)`
- `T & at (SZ_i1, SZ_i2, SZ_i3)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7, SZ_i8)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7, SZ_i8, SZ_i9)`
- `T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7, SZ_i8, SZ_i9, SZ_i10)`
- `const T & at (SZ_i1) const`
- `const T & at (SZ_i1, SZ_i2) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7, SZ_i8) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7, SZ_i8, SZ_i9) const`
- `const T & at (SZ_i1, SZ_i2, SZ_i3, SZ_i4, SZ_i5, SZ_i6, SZ_i7, SZ_i8, SZ_i9, SZ_i10) const`
- `array_iterator< array_base< T, SZ, N > > begin ()`
- `array_iterator< array_base< T, SZ, N > > end ()`
- `const_array_iterator< array_base< T, SZ, N > > begin () const`
- `const_array_iterator< array_base< T, SZ, N > > end () const`

Protected Attributes

- `::coder::detail::data_ptr< T, SZ > data_`
- `SZ size_ [N]`

Private Member Functions

- `void ensureCapacity (SZ_newNumel)`

6.5.1 Detailed Description

```
template<typename T, typename SZ, int N>
class coder::array_base< T, SZ, N >
```

Definition at line 461 of file coder_array.h.

6.5.2 Member Typedef Documentation

6.5.2.1 size_type

```
template<typename T , typename SZ , int N>
typedef SZ coder::array_base< T, SZ, N >::size_type
```

Definition at line 464 of file coder_array.h.

6.5.2.2 value_type

```
template<typename T , typename SZ , int N>
typedef T coder::array_base< T, SZ, N >::value_type
```

Definition at line 463 of file coder_array.h.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 array_base() [1/2]

```
template<typename T , typename SZ , int N>
coder::array_base< T, SZ, N >::array_base ( ) [inline]
```

Definition at line 466 of file coder_array.h.

```
466           {
467             std::memset(size_, 0, sizeof(SZ) * N);
468         }
```

6.5.3.2 array_base() [2/2]

```
template<typename T , typename SZ , int N>
coder::array_base< T, SZ, N >::array_base (
    T * _data,
    const SZ * _sz ) [inline]
```

Definition at line 470 of file coder_array.h.

```
471           : data_(_data, ::coder::detail::numel<N>::compute(_sz)) {
472             std::copy(_sz, _sz + N, size_);
473         }
```

6.5.4 Member Function Documentation

6.5.4.1 at() [1/20]

```
template<typename T, typename SZ, int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1 ) [inline]
```

Definition at line 847 of file coder_array.h.

```
847     {
848         ::coder::detail::match_dimensions<N == 1>::check ();
849         return data_[_i1];
850     }
```

6.5.4.2 at() [2/20]

```
template<typename T, typename SZ, int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1 ) const [inline]
```

Definition at line 888 of file coder_array.h.

```
888     {
889         ::coder::detail::match_dimensions<N == 1>::check ();
890         return data_[_i1];
891     }
```

6.5.4.3 at() [3/20]

```
template<typename T, typename SZ, int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2 ) [inline]
```

Definition at line 851 of file coder_array.h.

```
851     {
852         ::coder::detail::match_dimensions<N == 2>::check ();
853         return data_[index(_i1, _i2)];
854     }
```

6.5.4.4 at() [4/20]

```
template<typename T, typename SZ, int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2 ) const [inline]
```

Definition at line 892 of file coder_array.h.

```
892     {
893         ::coder::detail::match_dimensions<N == 2>::check ();
894         return data_[index(_i1, _i2)];
895     }
```

6.5.4.5 at() [5/20]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3 ) [inline]

Definition at line 855 of file coder_array.h.
855     {
856         ::coder::detail::match_dimensions<N == 3>::check();
857         return data_[index(_i1, _i2, _i3)];
858     }
```

6.5.4.6 at() [6/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3 ) const [inline]

Definition at line 896 of file coder_array.h.
896     {
897         ::coder::detail::match_dimensions<N == 3>::check();
898         return data_[index(_i1, _i2, _i3)];
899     }
```

6.5.4.7 at() [7/20]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4 ) [inline]

Definition at line 859 of file coder_array.h.
859     {
860         ::coder::detail::match_dimensions<N == 4>::check();
861         return data_[index(_i1, _i2, _i3, _i4)];
862     }
```

6.5.4.8 at() [8/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4 ) const [inline]

Definition at line 900 of file coder_array.h.
900     {
901         ::coder::detail::match_dimensions<N == 4>::check();
902         return data_[index(_i1, _i2, _i3, _i4)];
903     }
```

6.5.4.9 at() [9/20]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5 )  [inline]
```

Definition at line 863 of file coder_array.h.

```
863     {
864         ::coder::detail::match_dimensions<N == 5>::check();
865         return data_[index(_i1, _i2, _i3, _i4, _i5)];
866     }
```

6.5.4.10 at() [10/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5 ) const [inline]
```

Definition at line 904 of file coder_array.h.

```
904     {
905         ::coder::detail::match_dimensions<N == 5>::check();
906         return data_[index(_i1, _i2, _i3, _i4, _i5)];
907     }
```

6.5.4.11 at() [11/20]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6 )  [inline]
```

Definition at line 867 of file coder_array.h.

```
867     {
868         ::coder::detail::match_dimensions<N == 6>::check();
869         return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
870     }
```

6.5.4.12 at() [12/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6 ) const [inline]
```

Definition at line 908 of file coder_array.h.

```
908
909     ::coder::detail::match_dimensions<N == 6>::check();
910     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6)];
911 }
```

6.5.4.13 at() [13/20]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7 ) [inline]
```

Definition at line 871 of file coder_array.h.

```
871
872     ::coder::detail::match_dimensions<N == 7>::check();
873     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
874 }
```

6.5.4.14 at() [14/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7 ) const [inline]
```

Definition at line 912 of file coder_array.h.

```
912
913     ::coder::detail::match_dimensions<N == 7>::check();
914     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7)];
915 }
```

6.5.4.15 at() [15/20]

```
template<typename T, typename SZ, int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7,
    SZ _i8 )  [inline]
```

Definition at line 875 of file coder_array.h.

```
875
876     ::coder::detail::match_dimensions<N == 8>::check();
877     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
878 }
```

6.5.4.16 at() [16/20]

```
template<typename T, typename SZ, int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7,
    SZ _i8 ) const [inline]
```

Definition at line 916 of file coder_array.h.

```
916
917     ::coder::detail::match_dimensions<N == 8>::check();
918     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8)];
919 }
```

6.5.4.17 at() [17/20]

```
template<typename T, typename SZ, int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7,
    SZ _i8,
    SZ _i9 )  [inline]
```

Definition at line 879 of file coder_array.h.

```
879
880     ::coder::detail::match_dimensions<N == 9>::check();
881     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
882 }
```

6.5.4.18 at() [18/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7,
    SZ _i8,
    SZ _i9 ) const [inline]
```

Definition at line 920 of file coder_array.h.

```
920
921     ::coder::detail::match_dimensions<N == 9>::check();
922     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9)];
923 }
```

6.5.4.19 at() [19/20]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7,
    SZ _i8,
    SZ _i9,
    SZ _i10 ) [inline]
```

Definition at line 883 of file coder_array.h.

```
883
884     ::coder::detail::match_dimensions<N == 10>::check();
885     return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
886 }
```

6.5.4.20 at() [20/20]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::at (
    SZ _i1,
    SZ _i2,
    SZ _i3,
    SZ _i4,
    SZ _i5,
    SZ _i6,
    SZ _i7,
    SZ _i8,
```

```
    SZ _i9,
    SZ _i10 ) const [inline]
```

Definition at line 924 of file coder_array.h.

```
925         {
926             ::coder::detail::match_dimensions<N == 10>::check();
927             return data_[index(_i1, _i2, _i3, _i4, _i5, _i6, _i7, _i8, _i9, _i10)];
928         }
```

6.5.4.21 begin() [1/2]

```
template<typename T , typename SZ , int N>
array_iterator<array_base<T, SZ, N> > coder::array_base< T, SZ, N >::begin ( ) [inline]
```

Definition at line 930 of file coder_array.h.

```
930         {
931             return array_iterator<array_base<T, SZ, N> >(this, 0);
932         }
```

6.5.4.22 begin() [2/2]

```
template<typename T , typename SZ , int N>
const_array_iterator<array_base<T, SZ, N> > coder::array_base< T, SZ, N >::begin ( ) const
[inline]
```

Definition at line 936 of file coder_array.h.

```
936         {
937             return const_array_iterator<array_base<T, SZ, N> >(this, 0);
938         }
```

6.5.4.23 capacity()

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::capacity ( ) const [inline]
```

Definition at line 595 of file coder_array.h.

```
595         {
596             return data_.capacity();
597         }
```

6.5.4.24 clear()

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::clear ( ) [inline]
```

Definition at line 771 of file coder_array.h.

```
771         {
772             data_.clear();
773         }
```

6.5.4.25 data() [1/2]

```
template<typename T , typename SZ , int N>
T* coder::array_base< T, SZ, N >::data ( )  [inline]
```

Definition at line 775 of file coder_array.h.

```
775         {
776             return data_;
777 }
```

6.5.4.26 data() [2/2]

```
template<typename T , typename SZ , int N>
const T* coder::array_base< T, SZ, N >::data ( ) const  [inline]
```

Definition at line 779 of file coder_array.h.

```
779         {
780             return data_;
781 }
```

6.5.4.27 end() [1/2]

```
template<typename T , typename SZ , int N>
array_iterator<array_base<T, SZ, N> > coder::array_base< T, SZ, N >::end ( )  [inline]
```

Definition at line 933 of file coder_array.h.

```
933         {
934             return array_iterator<array_base<T, SZ, N> >(this, this->numel());
935 }
```

6.5.4.28 end() [2/2]

```
template<typename T , typename SZ , int N>
const_array_iterator<array_base<T, SZ, N> > coder::array_base< T, SZ, N >::end ( ) const
[inline]
```

Definition at line 939 of file coder_array.h.

```
939         {
940             return const_array_iterator<array_base<T, SZ, N> >(this, this->numel());
941 }
```

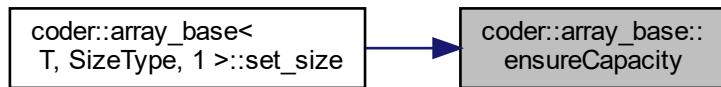
6.5.4.29 ensureCapacity()

```
template<typename T, typename SZ, int N>
void coder::array_base< T, SZ, N >::ensureCapacity (
    SZ _newNumel ) [inline], [private]
```

Definition at line 948 of file coder_array.h.

```
948         if (_newNumel > data_.capacity()) {
949             SZ i = data_.capacity();
950             if (i < 16) {
951                 i = 16;
952             }
953         }
954         while (i < _newNumel) {
955             if (i > 1073741823) {
956                 i = MAX_int32_T;
957             } else {
958                 i <<= 1;
959             }
960         }
961         data_.reserve(i);
962     }
963     data_.resize(_newNumel);
964 }
```

Here is the caller graph for this function:



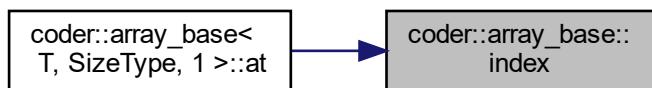
6.5.4.30 index() [1/10]

```
template<typename T, typename SZ, int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1 ) const [inline]
```

Definition at line 795 of file coder_array.h.

```
795         {
796             ::coder::detail::match_dimensions<N == 1>::check();
797             const SZ indices[] = {_n1};
798             return ::coder::detail::index_nd<1>::compute(size_, indices);
799         }
```

Here is the caller graph for this function:



6.5.4.31 `index()` [2/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2 ) const [inline]

Definition at line 800 of file coder_array.h.
800      {
801          ::coder::detail::match_dimensions<N == 2>::check();
802          const SZ indices[] = {_n1, _n2};
803          return ::coder::detail::index_nd<2>::compute(size_, indices);
804      }
```

6.5.4.32 `index()` [3/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3 ) const [inline]

Definition at line 805 of file coder_array.h.
805      {
806          ::coder::detail::match_dimensions<N == 3>::check();
807          const SZ indices[] = {_n1, _n2, _n3};
808          return ::coder::detail::index_nd<3>::compute(size_, indices);
809      }
```

6.5.4.33 `index()` [4/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4 ) const [inline]

Definition at line 810 of file coder_array.h.
810      {
811          ::coder::detail::match_dimensions<N == 4>::check();
812          const SZ indices[] = {_n1, _n2, _n3, _n4};
813          return ::coder::detail::index_nd<4>::compute(size_, indices);
814      }
```

6.5.4.34 index() [5/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5 ) const [inline]
```

Definition at line 815 of file coder_array.h.

```
815
816     ::coder::detail::match_dimensions<N == 5>::check();
817     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5};
818     return ::coder::detail::index_nd<5>::compute(size_, indices);
819 }
```

6.5.4.35 index() [6/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6 ) const [inline]
```

Definition at line 820 of file coder_array.h.

```
820
821     ::coder::detail::match_dimensions<N == 6>::check();
822     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6};
823     return ::coder::detail::index_nd<6>::compute(size_, indices);
824 }
```

6.5.4.36 index() [7/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7 ) const [inline]
```

Definition at line 825 of file coder_array.h.

```
825
826     ::coder::detail::match_dimensions<N == 7>::check();
827     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
828     return ::coder::detail::index_nd<7>::compute(size_, indices);
829 }
```

6.5.4.37 index() [8/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8 ) const [inline]
```

Definition at line 830 of file coder_array.h.

```
830
831     ::coder::detail::match_dimensions<N == 8>::check();
832     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
833     return ::coder::detail::index_nd<8>::compute(size_, indices);
834 }
```

6.5.4.38 index() [9/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
    SZ _n9 ) const [inline]
```

Definition at line 835 of file coder_array.h.

```
835
836     ::coder::detail::match_dimensions<N == 9>::check();
837     const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
838     return ::coder::detail::index_nd<9>::compute(size_, indices);
839 }
```

6.5.4.39 index() [10/10]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::index (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
```

```
    SZ _n9,
    SZ _n10 ) const [inline]
```

Definition at line 840 of file coder_array.h.

```
841     {
842         ::coder::detail::match_dimensions<N == 10>::check();
843         const SZ indices[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
844         return ::coder::detail::index_nd<10>::compute(size_, indices);
845     }
```

6.5.4.40 is_owner()

```
template<typename T , typename SZ , int N>
bool coder::array_base< T, SZ, N >::is_owner ( ) const [inline]
```

Definition at line 587 of file coder_array.h.

```
587     {
588         return data_.is_owner();
589     }
```

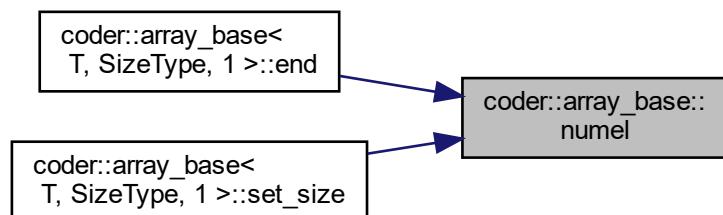
6.5.4.41 numel()

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::numel ( ) const [inline]
```

Definition at line 791 of file coder_array.h.

```
791     {
792         return ::coder::detail::numel<N>::compute(size_);
793     }
```

Here is the caller graph for this function:



6.5.4.42 operator=()

```
template<typename T , typename SZ , int N>
array_base& coder::array_base< T, SZ, N >::operator= (
    const array_base< T, SZ, N > & _other ) [inline]
```

Definition at line 475 of file coder_array.h.

```
475
476     data_.copy(_other.data_);
477     std::copy(_other.size_, _other.size_ + N, size_);
478     return *this;
479 }
```

6.5.4.43 operator[]() [1/2]

```
template<typename T , typename SZ , int N>
T& coder::array_base< T, SZ, N >::operator[] (
    SZ _index ) [inline]
```

Definition at line 763 of file coder_array.h.

```
763
764     return data_[_index];
765 }
```

6.5.4.44 operator[]() [2/2]

```
template<typename T , typename SZ , int N>
const T& coder::array_base< T, SZ, N >::operator[] (
    SZ _index ) const [inline]
```

Definition at line 767 of file coder_array.h.

```
767
768     return data_[_index];
769 }
```

6.5.4.45 reshape() [1/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 1> coder::array_base< T, SZ, N >::reshape (
    SZ _n1 ) const [inline]
```

Definition at line 710 of file coder_array.h.

```
710
711     const SZ ns[] = {_n1};
712     return reshape_n(ns);
713 }
```

6.5.4.46 `reshape()` [2/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 2> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2 ) const [inline]
```

Definition at line 715 of file `coder_array.h`.

```
715
716     const SZ ns[] = {_n1, _n2};
717     return reshape_n(ns);
718 }
```

6.5.4.47 `reshape()` [3/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 3> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3 ) const [inline]
```

Definition at line 720 of file `coder_array.h`.

```
720
721     const SZ ns[] = {_n1, _n2, _n3};
722     return reshape_n(ns);
723 }
```

6.5.4.48 `reshape()` [4/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 4> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4 ) const [inline]
```

Definition at line 725 of file `coder_array.h`.

```
725
726     const SZ ns[] = {_n1, _n2, _n3, _n4};
727     return reshape_n(ns);
728 }
```

6.5.4.49 `reshape()` [5/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 5> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5 ) const [inline]
```

Definition at line 730 of file `coder_array.h`.

```
730
731     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5};
732     return reshape_n(ns);
733 }
```

6.5.4.50 reshape() [6/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 6> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6 ) const [inline]
```

Definition at line 735 of file coder_array.h.

```
735
736     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6};
737     return reshape_n(ns);
738 }
```

6.5.4.51 reshape() [7/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 7> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7 ) const [inline]
```

Definition at line 740 of file coder_array.h.

```
740
741     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7};
742     return reshape_n(ns);
743 }
```

6.5.4.52 reshape() [8/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 8> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8 ) const [inline]
```

Definition at line 745 of file coder_array.h.

```
746     {
747     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8};
748     return reshape_n(ns);
749 }
```

6.5.4.53 `reshape()` [9/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 9> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
    SZ _n9 ) const [inline]
```

Definition at line 752 of file `coder_array.h`.

```
752
753     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9};
754     return reshape_n(ns);
755 }
```

6.5.4.54 `reshape()` [10/10]

```
template<typename T , typename SZ , int N>
array_base<T, SZ, 10> coder::array_base< T, SZ, N >::reshape (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
    SZ _n9,
    SZ _n10 ) const [inline]
```

Definition at line 758 of file `coder_array.h`.

```
758
759     const SZ ns[] = {_n1, _n2, _n3, _n4, _n5, _n6, _n7, _n8, _n9, _n10};
760     return reshape_n(ns);
761 }
```

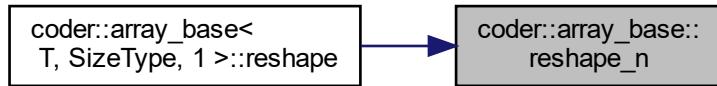
6.5.4.55 `reshape_n()`

```
template<typename T , typename SZ , int N>
template<size_t N1>
array_base<T, SZ, N1> coder::array_base< T, SZ, N >::reshape_n (
    const SZ(&) _ns[N1] ) const [inline]
```

Definition at line 705 of file `coder_array.h`.

```
705
706     array_base<T, SZ, N1> reshaped(const_cast<T*>(&data_[0]), _ns);
707     return reshaped;
708 }
```

Here is the caller graph for this function:



6.5.4.56 set() [1/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1 )  [inline]
```

Definition at line 481 of file coder_array.h.

```
481
482     ::coder::detail::match_dimensions<N == 1>::check();
483     data_.set(_data, _n1);
484     size_[0] = _n1;
485 }
```

6.5.4.57 set() [2/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2 )  [inline]
```

Definition at line 487 of file coder_array.h.

```
487
488     ::coder::detail::match_dimensions<N == 2>::check();
489     data_.set(_data, _n1 * _n2);
490     size_[0] = _n1;
491     size_[1] = _n2;
492 }
```

6.5.4.58 set() [3/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3 )  [inline]
```

Definition at line 494 of file coder_array.h.

```
494      ::coder::detail::match_dimensions<N == 3>::check();
495      data_.set(_data, _n1 * _n2 * _n3);
496      size_[0] = _n1;
497      size_[1] = _n2;
498      size_[2] = _n3;
499
500 }
```

6.5.4.59 set() [4/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4 )  [inline]
```

Definition at line 502 of file coder_array.h.

```
502      ::coder::detail::match_dimensions<N == 4>::check();
503      data_.set(_data, _n1 * _n2 * _n3 * _n4);
504      size_[0] = _n1;
505      size_[1] = _n2;
506      size_[2] = _n3;
507      size_[3] = _n4;
508
509 }
```

6.5.4.60 set() [5/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5 )  [inline]
```

Definition at line 511 of file coder_array.h.

```
511      ::coder::detail::match_dimensions<N == 5>::check();
512      data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5);
513      size_[0] = _n1;
514      size_[1] = _n2;
515      size_[2] = _n3;
516      size_[3] = _n4;
517      size_[4] = _n5;
518
519 }
```

6.5.4.61 set() [6/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6 )  [inline]
```

Definition at line 521 of file coder_array.h.

```
521
522     ::coder::detail::match_dimensions<N == 6>::check();
523     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6);
524     size_[0] = _n1;
525     size_[1] = _n2;
526     size_[2] = _n3;
527     size_[3] = _n4;
528     size_[4] = _n5;
529     size_[5] = _n6;
530 }
```

6.5.4.62 set() [7/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7 )  [inline]
```

Definition at line 532 of file coder_array.h.

```
532
533     ::coder::detail::match_dimensions<N == 7>::check();
534     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7);
535     size_[0] = _n1;
536     size_[1] = _n2;
537     size_[2] = _n3;
538     size_[3] = _n4;
539     size_[4] = _n5;
540     size_[5] = _n6;
541     size_[6] = _n7;
542 }
```

6.5.4.63 set() [8/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
```

```

SZ _n3,
SZ _n4,
SZ _n5,
SZ _n6,
SZ _n7,
SZ _n8 ) [inline]

```

Definition at line 544 of file coder_array.h.

```

544
545     ::coder::detail::match_dimensions<N == 8>::check();
546     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8);
547     size_[0] = _n1;
548     size_[1] = _n2;
549     size_[2] = _n3;
550     size_[3] = _n4;
551     size_[4] = _n5;
552     size_[5] = _n6;
553     size_[6] = _n7;
554     size_[7] = _n8;
555 }

```

6.5.4.64 set() [9/10]

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
    SZ _n9 ) [inline]

```

Definition at line 557 of file coder_array.h.

```

557
558     ::coder::detail::match_dimensions<N == 9>::check();
559     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9);
560     size_[0] = _n1;
561     size_[1] = _n2;
562     size_[2] = _n3;
563     size_[3] = _n4;
564     size_[4] = _n5;
565     size_[5] = _n6;
566     size_[6] = _n7;
567     size_[7] = _n8;
568     size_[8] = _n9;
569 }

```

6.5.4.65 set() [10/10]

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set (
    T * _data,
    SZ _n1,
    SZ _n2,
    SZ _n3,

```

```

SZ _n4,
SZ _n5,
SZ _n6,
SZ _n7,
SZ _n8,
SZ _n9,
SZ _n10 ) [inline]

```

Definition at line 572 of file coder_array.h.

```

572
573     ::coder::detail::match_dimensions<N == 10>::check();
574     data_.set(_data, _n1 * _n2 * _n3 * _n4 * _n5 * _n6 * _n7 * _n8 * _n9 * _n10);
575     size_[0] = _n1;
576     size_[1] = _n2;
577     size_[2] = _n3;
578     size_[3] = _n4;
579     size_[4] = _n5;
580     size_[5] = _n6;
581     size_[6] = _n7;
582     size_[7] = _n8;
583     size_[8] = _n9;
584     size_[9] = _n10;
585 }

```

6.5.4.66 set_owner()

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_owner (
    bool b ) [inline]

```

Definition at line 591 of file coder_array.h.

```

591 {
592     data_.set_owner(b);
593 }

```

6.5.4.67 set_size() [1/10]

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1 ) [inline]

```

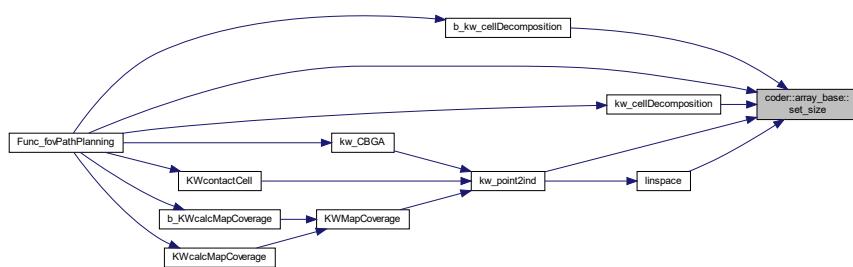
Definition at line 599 of file coder_array.h.

```

599
600     ::coder::detail::match_dimensions<N == 1>::check();
601     size_[0] = _n1;
602     ensureCapacity(numel());
603 }

```

Here is the caller graph for this function:



6.5.4.68 set_size() [2/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2 )  [inline]
```

Definition at line 605 of file coder_array.h.

```
605      {
606          ::coder::detail::match_dimensions<N == 2>::check ();
607          size_[0] = _n1;
608          size_[1] = _n2;
609          ensureCapacity(numel ());
610      }
```

6.5.4.69 set_size() [3/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3 )  [inline]
```

Definition at line 612 of file coder_array.h.

```
612      {
613          ::coder::detail::match_dimensions<N == 3>::check ();
614          size_[0] = _n1;
615          size_[1] = _n2;
616          size_[2] = _n3;
617          ensureCapacity(numel ());
618      }
```

6.5.4.70 set_size() [4/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4 )  [inline]
```

Definition at line 620 of file coder_array.h.

```
620      {
621          ::coder::detail::match_dimensions<N == 4>::check ();
622          size_[0] = _n1;
623          size_[1] = _n2;
624          size_[2] = _n3;
625          size_[3] = _n4;
626          ensureCapacity(numel ());
627      }
```

6.5.4.71 set_size() [5/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5 )  [inline]
```

Definition at line 629 of file coder_array.h.

```
629
630     ::coder::detail::match_dimensions<N == 5>::check(); {
631     size_[0] = _n1;
632     size_[1] = _n2;
633     size_[2] = _n3;
634     size_[3] = _n4;
635     size_[4] = _n5;
636     ensureCapacity(numel());
637 }
```

6.5.4.72 set_size() [6/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6 )  [inline]
```

Definition at line 639 of file coder_array.h.

```
639
640     ::coder::detail::match_dimensions<N == 6>::check(); {
641     size_[0] = _n1;
642     size_[1] = _n2;
643     size_[2] = _n3;
644     size_[3] = _n4;
645     size_[4] = _n5;
646     size_[5] = _n6;
647     ensureCapacity(numel());
648 }
```

6.5.4.73 set_size() [7/10]

```
template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7 )  [inline]
```

Definition at line 650 of file coder_array.h.

```

650
651     ::coder::detail::match_dimensions<N == 7>::check();
652     size_[0] = _n1;
653     size_[1] = _n2;
654     size_[2] = _n3;
655     size_[3] = _n4;
656     size_[4] = _n5;
657     size_[5] = _n6;
658     size_[6] = _n7;
659     ensureCapacity(numel());
660 }

```

6.5.4.74 set_size() [8/10]

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8 ) [inline]

```

Definition at line 662 of file coder_array.h.

```

662
663     ::coder::detail::match_dimensions<N == 8>::check();
664     size_[0] = _n1;
665     size_[1] = _n2;
666     size_[2] = _n3;
667     size_[3] = _n4;
668     size_[4] = _n5;
669     size_[5] = _n6;
670     size_[6] = _n7;
671     size_[7] = _n8;
672     ensureCapacity(numel());
673 }

```

6.5.4.75 set_size() [9/10]

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
    SZ _n9 ) [inline]

```

Definition at line 675 of file coder_array.h.

```

675
676     ::coder::detail::match_dimensions<N == 9>::check();
677     size_[0] = _n1;
678     size_[1] = _n2;
679     size_[2] = _n3;
680     size_[3] = _n4;
681     size_[4] = _n5;

```

```

682     size_[5] = _n6;
683     size_[6] = _n7;
684     size_[7] = _n8;
685     size_[8] = _n9;
686     ensureCapacity(numel());
687 }

```

6.5.4.76 set_size() [10/10]

```

template<typename T , typename SZ , int N>
void coder::array_base< T, SZ, N >::set_size (
    SZ _n1,
    SZ _n2,
    SZ _n3,
    SZ _n4,
    SZ _n5,
    SZ _n6,
    SZ _n7,
    SZ _n8,
    SZ _n9,
    SZ _n10 ) [inline]

```

Definition at line 689 of file coder_array.h.

```

689
690     ::coder::detail::match_dimensions<N == 10>::check();
691     size_[0] = _n1;
692     size_[1] = _n2;
693     size_[2] = _n3;
694     size_[3] = _n4;
695     size_[4] = _n5;
696     size_[5] = _n6;
697     size_[6] = _n7;
698     size_[7] = _n8;
699     size_[8] = _n9;
700     size_[9] = _n10;
701     ensureCapacity(numel());
702 }

```

6.5.4.77 size() [1/2]

```

template<typename T , typename SZ , int N>
const SZ* coder::array_base< T, SZ, N >::size ( ) const [inline]

```

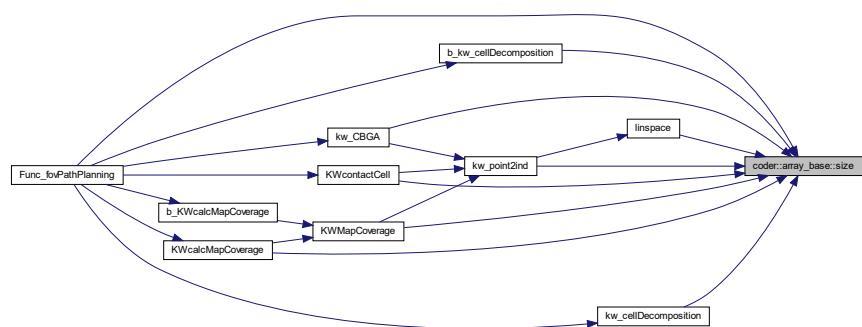
Definition at line 783 of file coder_array.h.

```

783
784     return &size_[0];
785 }

```

Here is the caller graph for this function:



6.5.4.78 `size()` [2/2]

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::size (
    SZ _index ) const [inline]
```

Definition at line 787 of file coder_array.h.

```
787
788     return size_[_index];
789 }
```

6.5.5 Member Data Documentation

6.5.5.1 `data_`

```
template<typename T , typename SZ , int N>
::coder::detail::data_ptr<T, SZ> coder::array_base< T, SZ, N >::data_ [protected]
```

Definition at line 944 of file coder_array.h.

6.5.5.2 `size_`

```
template<typename T , typename SZ , int N>
SZ coder::array_base< T, SZ, N >::size_[N] [protected]
```

Definition at line 945 of file coder_array.h.

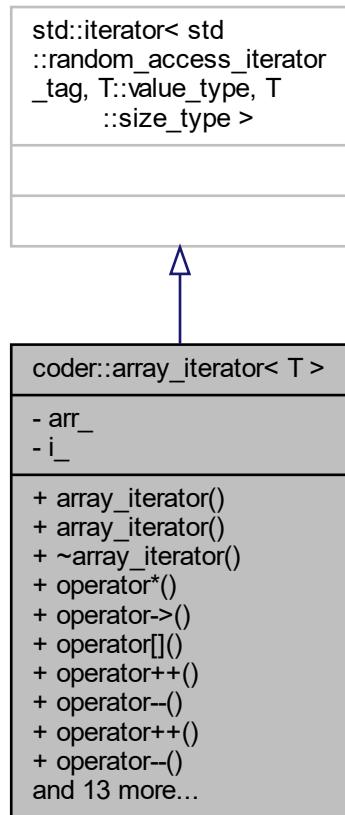
The documentation for this class was generated from the following file:

- codegen/lib(Func_fovPathPlanning/coder_array.h

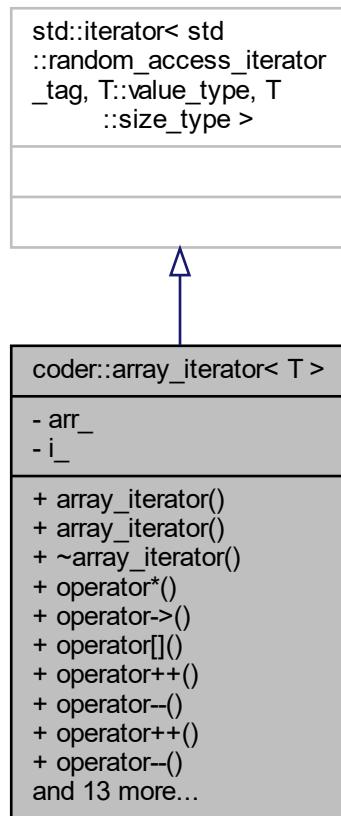
6.6 coder::array_iterator< T > Class Template Reference

```
#include <coder_array.h>
```

Inheritance diagram for coder::array_iterator< T >:



Collaboration diagram for coder::array_iterator< T >:



Public Member Functions

- `array_iterator ()`
- `array_iterator (const array_iterator< T > &other)`
- `~array_iterator ()`
- `T::value_type & operator* () const`
- `T::value_type * operator-> () const`
- `T::value_type & operator[] (typename T::size_type _di) const`
- `array_iterator< T > & operator++ ()`
- `array_iterator< T > & operator-- ()`
- `array_iterator< T > operator++ (int)`
- `array_iterator< T > operator-- (int)`
- `array_iterator< T > & operator= (const array_iterator< T > &_other)`
- `bool operator== (const array_iterator< T > &_other) const`
- `bool operator!= (const array_iterator< T > &_other) const`
- `bool operator< (const array_iterator< T > &_other) const`
- `bool operator> (const array_iterator< T > &_other) const`
- `bool operator<= (const array_iterator< T > &_other) const`
- `bool operator>= (const array_iterator< T > &_other) const`
- `array_iterator< T > operator+ (typename T::size_type _add) const`

- `array_iterator< T > & operator+= (typename T::size_type _add)`
- `array_iterator< T > operator- (typename T::size_type _subtract) const`
- `array_iterator< T > & operator-= (typename T::size_type _subtract)`
- `T::size_type operator- (const array_iterator< T > &_other) const`
- `array_iterator (T *_arr, typename T::size_type _i)`

Private Attributes

- `T * arr_`
- `T::size_type i_`

6.6.1 Detailed Description

```
template<typename T>
class coder::array_iterator< T >
```

Definition at line 212 of file coder_array.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 array_iterator() [1/3]

```
template<typename T >
coder::array_iterator< T >::array_iterator ( ) [inline]
```

Definition at line 216 of file coder_array.h.

```
217     : arr_(NULL)
218     , i_(0) {
219 }
```

6.6.2.2 array_iterator() [2/3]

```
template<typename T >
coder::array_iterator< T >::array_iterator (
    const array_iterator< T > & other ) [inline]
```

Definition at line 220 of file coder_array.h.

```
221     : arr_(other.arr_)
222     , i_(other.i_) {
223 }
```

6.6.2.3 ~array_iterator()

```
template<typename T >
coder::array_iterator< T >::~array_iterator ( ) [inline]
```

Definition at line 224 of file coder_array.h.

```
224     {
225 }
```

6.6.2.4 array_iterator() [3/3]

```
template<typename T >
coder::array_iterator< T >::array_iterator (
    T * _arr,
    typename T::size_type _i ) [inline]
```

Definition at line 297 of file coder_array.h.

```
298     : arr_(_arr)
299     , i_(_i) {
300 }
```

6.6.3 Member Function Documentation

6.6.3.1 operator"!=()

```
template<typename T >
bool coder::array_iterator< T >::operator!= (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 260 of file coder_array.h.

```
260
261     return i_ != _other.i_;
262 }
```

6.6.3.2 operator*()

```
template<typename T >
T::value_type& coder::array_iterator< T >::operator* ( ) const [inline]
```

Definition at line 226 of file coder_array.h.

```
226
227     return (*arr_)[i_];
228 }
```

6.6.3.3 operator+()

```
template<typename T >
array_iterator<T> coder::array_iterator< T >::operator+ (
    typename T::size_type _add ) const [inline]
```

Definition at line 275 of file coder_array.h.

```
275     array_iterator<T> cp(*this); {  
276         cp.i_ += _add;  
277         return cp;  
278     }
```

6.6.3.4 operator++() [1/2]

```
template<typename T >
array_iterator<T>& coder::array_iterator< T >::operator++ ( ) [inline]
```

Definition at line 235 of file coder_array.h.

```
235     {  
236         ++i_;  
237         return *this;  
238     }
```

6.6.3.5 operator++() [2/2]

```
template<typename T >
array_iterator<T> coder::array_iterator< T >::operator++ ( int ) [inline]
```

Definition at line 243 of file coder_array.h.

```
243     {  
244         array_iterator<T> cp(*this);  
245         ++i_;  
246         return cp;  
247     }
```

6.6.3.6 operator+=()

```
template<typename T >
array_iterator<T>& coder::array_iterator< T >::operator+= ( typename T::size_type _add ) [inline]
```

Definition at line 280 of file coder_array.h.

```
280     {  
281         this->i_ += _add;  
282         return *this;  
283     }
```

6.6.3.7 operator-() [1/2]

```
template<typename T >
T::size_type coder::array_iterator< T >::operator- (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 293 of file coder_array.h.

```
293
294     return static_cast<typename T::size_type>(this->i_ - _other.i_);
295 }
```

6.6.3.8 operator-() [2/2]

```
template<typename T >
array_iterator<T> coder::array_iterator< T >::operator- (
    typename T::size_type _subtract ) const [inline]
```

Definition at line 284 of file coder_array.h.

```
284
285     array_iterator<T> cp(*this);
286     cp.i_ -= _subtract;
287     return cp;
288 }
```

6.6.3.9 operator--() [1/2]

```
template<typename T >
array_iterator<T>& coder::array_iterator< T >::operator-- ( ) [inline]
```

Definition at line 239 of file coder_array.h.

```
239
240     --i_;
241     return *this;
242 }
```

6.6.3.10 operator--() [2/2]

```
template<typename T >
array_iterator<T> coder::array_iterator< T >::operator-- (
    int ) [inline]
```

Definition at line 248 of file coder_array.h.

```
248
249     array_iterator<T> cp(*this);
250     --i_;
251     return cp;
252 }
```

6.6.3.11 operator-=()

```
template<typename T >
array_iterator<T>& coder::array_iterator< T >::operator-= (
    typename T::size_type _subtract ) [inline]
```

Definition at line 289 of file coder_array.h.

```
289
290     this->i_ -= _subtract;
291     return *this;
292 }
```

6.6.3.12 operator->()

```
template<typename T >
T::value_type* coder::array_iterator< T >::operator-> () const [inline]
```

Definition at line 229 of file coder_array.h.

```
229
230     return &(*arr_)[i_];
231 }
```

6.6.3.13 operator<()

```
template<typename T >
bool coder::array_iterator< T >::operator< (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 263 of file coder_array.h.

```
263
264     return i_ < _other.i_;
265 }
```

6.6.3.14 operator<=()

```
template<typename T >
bool coder::array_iterator< T >::operator<= (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 269 of file coder_array.h.

```
269
270     return i_ <= _other.i_;
271 }
```

6.6.3.15 operator=()

```
template<typename T >
array_iterator<T>& coder::array_iterator< T >::operator= (
    const array_iterator< T > & _other ) [inline]
```

Definition at line 253 of file coder_array.h.

```
253
254     this->i_ = _other.i_;
255     return *this;
256 }
```

6.6.3.16 operator==()

```
template<typename T >
bool coder::array_iterator< T >::operator== (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 257 of file coder_array.h.

```
257
258     return i_ == _other.i_;
259 }
```

6.6.3.17 operator>()

```
template<typename T >
bool coder::array_iterator< T >::operator> (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 266 of file coder_array.h.

```
266
267     return i_ > _other.i_;
268 }
```

6.6.3.18 operator>=()

```
template<typename T >
bool coder::array_iterator< T >::operator>= (
    const array_iterator< T > & _other ) const [inline]
```

Definition at line 272 of file coder_array.h.

```
272
273     return i_ >= _other.i_;
274 }
```

6.6.3.19 operator[]()

```
template<typename T >
T::value_type& coder::array_iterator< T >::operator[] (
    typename T::size_type _di ) const [inline]
```

Definition at line 232 of file coder_array.h.

```
232
233     return (*arr_)[i_ + _di];
234 }
```

6.6.4 Member Data Documentation

6.6.4.1 arr_

```
template<typename T >
T* coder::array_iterator< T >::arr_ [private]
```

Definition at line 303 of file coder_array.h.

6.6.4.2 i_

```
template<typename T >
T::size_type coder::array_iterator< T >::i_ [private]
```

Definition at line 304 of file coder_array.h.

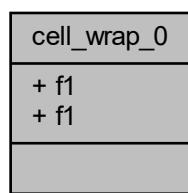
The documentation for this class was generated from the following file:

- codegen/lib(Func_fovPathPlanning/coder_array.h)

6.7 cell_wrap_0 Struct Reference

```
#include <Func_fovPathPlanning_types.h>
```

Collaboration diagram for cell_wrap_0:



Public Attributes

- double `f1` [10]
- real_T `f1` [10]

6.7.1 Detailed Description

Definition at line 26 of file Func_fovPathPlanning_types.h.

6.7.2 Member Data Documentation

6.7.2.1 `f1` [1/2]

```
double cell_wrap_0::f1[10]
```

Definition at line 28 of file Func_fovPathPlanning_types.h.

6.7.2.2 `f1` [2/2]

```
real_T cell_wrap_0::f1[10]
```

Definition at line 27 of file _coder_Func_fovPathPlanning_api.h.

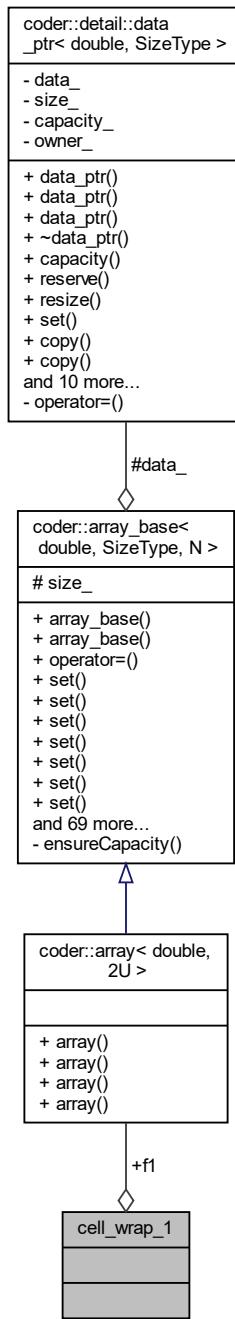
The documentation for this struct was generated from the following files:

- codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_types.h)
- codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.h)

6.8 cell_wrap_1 Struct Reference

```
#include <Func_fovPathPlanning_types.h>
```

Collaboration diagram for cell_wrap_1:



Public Attributes

- `coder::array< double, 2U > f1`

6.8.1 Detailed Description

Definition at line 31 of file Func_fovPathPlanning_types.h.

6.8.2 Member Data Documentation

6.8.2.1 f1

```
coder::array<double, 2U> cell_wrap_1::f1
```

Definition at line 33 of file Func_fovPathPlanning_types.h.

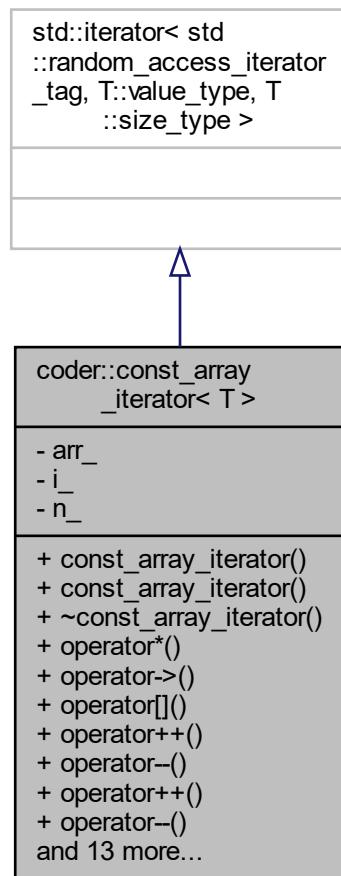
The documentation for this struct was generated from the following file:

- codegen/lib(Func_fovPathPlanning/[Func_fovPathPlanning_types.h](#)

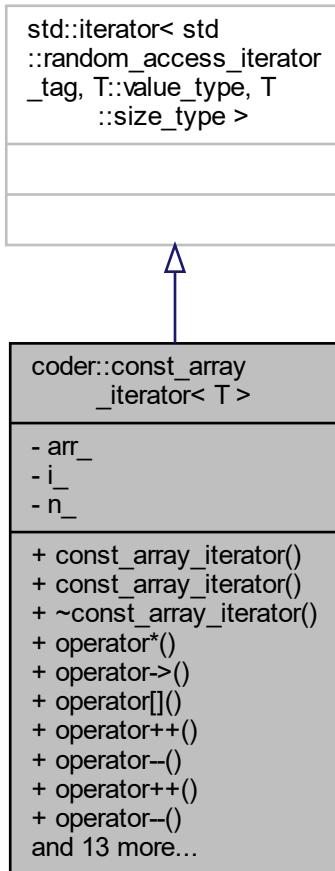
6.9 coder::const_array_iterator< T > Class Template Reference

```
#include <coder_array.h>
```

Inheritance diagram for coder::const_array_iterator< T >:



Collaboration diagram for coder::const_array_iterator< T >:



Public Member Functions

- `const_array_iterator ()`
- `const_array_iterator (const const_array_iterator< T > &other)`
- `~const_array_iterator ()`
- `const T::value_type & operator* () const`
- `const T::value_type * operator-> () const`
- `const T::value_type & operator[] (typename T::size_type _di) const`
- `const_array_iterator< T > & operator++ ()`
- `const_array_iterator< T > & operator-- ()`
- `const_array_iterator< T > operator++ (int)`
- `const_array_iterator< T > operator-- (int)`
- `const_array_iterator< T > & operator= (const const_array_iterator< T > &_other)`
- `bool operator== (const const_array_iterator< T > &_other) const`
- `bool operator!= (const const_array_iterator< T > &_other) const`
- `bool operator< (const const_array_iterator< T > &_other) const`
- `bool operator> (const const_array_iterator< T > &_other) const`
- `bool operator<= (const const_array_iterator< T > &_other) const`

- bool `operator>=` (const `const_array_iterator< T >` &`_other`) const
- `const_array_iterator< T >` `operator+` (typename `T::size_type` `_add`) const
- `const_array_iterator< T >` & `operator+=` (typename `T::size_type` `_add`)
- `const_array_iterator< T >` `operator-` (typename `T::size_type` `_subtract`) const
- `const_array_iterator< T >` & `operator-=` (typename `T::size_type` `_subtract`)
- `T::size_type` `operator-` (const `const_array_iterator< T >` &`_other`) const
- `const_array_iterator` (const `T * arr`, typename `T::size_type` `i`)

Private Attributes

- `const T * arr_`
- `T::size_type i_`
- `T::size_type n_`

6.9.1 Detailed Description

```
template<typename T>
class coder::const_array_iterator< T >
```

Definition at line 309 of file coder_array.h.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 const_array_iterator() [1/3]

```
template<typename T >
coder::const_array_iterator< T >::const_array_iterator ( ) [inline]
```

Definition at line 313 of file coder_array.h.

```
314     : arr_(NULL)
315     , i_(0) {
316 }
```

6.9.2.2 const_array_iterator() [2/3]

```
template<typename T >
coder::const_array_iterator< T >::const_array_iterator (
    const const_array_iterator< T > & other ) [inline]
```

Definition at line 317 of file coder_array.h.

```
318     : arr_(other.arr_)
319     , i_(other.i_) {
320 }
```

6.9.2.3 ~const_array_iterator()

```
template<typename T >
coder::const_array_iterator< T >::~const_array_iterator ( ) [inline]
```

Definition at line 321 of file coder_array.h.
 321 {
 322 }

6.9.2.4 const_array_iterator() [3/3]

```
template<typename T >
coder::const_array_iterator< T >::const_array_iterator (
    const T * _arr,
    typename T::size_type _i ) [inline]
```

Definition at line 396 of file coder_array.h.
 397 : arr_(_arr)
 398 , i_(_i) {
 399 }

6.9.3 Member Function Documentation

6.9.3.1 operator"!=()

```
template<typename T >
bool coder::const_array_iterator< T >::operator!= (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 357 of file coder_array.h.
 357 {
 358 return i_ != _other.i_;
 359 }

6.9.3.2 operator*()

```
template<typename T >
const T::value_type& coder::const_array_iterator< T >::operator* ( ) const [inline]
```

Definition at line 323 of file coder_array.h.
 323 {
 324 return (*arr_)[i_];
 325 }

6.9.3.3 operator+()

```
template<typename T >
const_array_iterator<T> coder::const_array_iterator< T >::operator+ (
    typename T::size_type _add ) const [inline]
```

Definition at line 372 of file coder_array.h.

```
372
373     const_array_iterator<T> cp(*this);
374     cp.i_ += _add;
375     return cp;
376 }
```

6.9.3.4 operator++() [1/2]

```
template<typename T >
const_array_iterator<T>& coder::const_array_iterator< T >::operator++ ( ) [inline]
```

Definition at line 332 of file coder_array.h.

```
332
333     ++i_;
334     return *this;
335 }
```

6.9.3.5 operator++() [2/2]

```
template<typename T >
const_array_iterator<T> coder::const_array_iterator< T >::operator++ ( int ) [inline]
```

Definition at line 340 of file coder_array.h.

```
340
341     const_array_iterator<T> copy(*this);
342     ++i_;
343     return copy;
344 }
```

6.9.3.6 operator+=()

```
template<typename T >
const_array_iterator<T>& coder::const_array_iterator< T >::operator+= ( typename T::size_type _add ) [inline]
```

Definition at line 377 of file coder_array.h.

```
377
378     this->i_ += _add;
379     return *this;
380 }
```

6.9.3.7 operator-() [1/2]

```
template<typename T >
T::size_type coder::const_array_iterator< T >::operator- (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 392 of file coder_array.h.

```
392
393     return static_cast<typename T::size_type>(this->i_ - _other.i_);
394 }
```

6.9.3.8 operator-() [2/2]

```
template<typename T >
const_array_iterator<T> coder::const_array_iterator< T >::operator- (
    typename T::size_type _subtract ) const [inline]
```

Definition at line 381 of file coder_array.h.

```
381
382     const_array_iterator<T> cp(*this);
383     cp.i_ -= _subtract;
384     return cp;
385 }
```

6.9.3.9 operator--() [1/2]

```
template<typename T >
const_array_iterator<T>& coder::const_array_iterator< T >::operator-- ( ) [inline]
```

Definition at line 336 of file coder_array.h.

```
336
337     --i_;
338     return *this;
339 }
```

6.9.3.10 operator--() [2/2]

```
template<typename T >
const_array_iterator<T> coder::const_array_iterator< T >::operator-- ( int ) [inline]
```

Definition at line 345 of file coder_array.h.

```
345
346     const_array_iterator copy(*this);
347     --i_;
348     return copy;
349 }
```

6.9.3.11 operator-=()

```
template<typename T >
const_array_iterator<T>& coder::const_array_iterator< T >::operator-= (
    typename T::size_type _subtract ) [inline]
```

Definition at line 387 of file coder_array.h.

```
387
388     this->i_ -= _subtract;
389     return *this;
390 }
```

6.9.3.12 operator->()

```
template<typename T >
const T::value_type* coder::const_array_iterator< T >::operator-> () const [inline]
```

Definition at line 326 of file coder_array.h.

```
326
327     return &(*arr_)[i_];
328 }
```

6.9.3.13 operator<()

```
template<typename T >
bool coder::const_array_iterator< T >::operator< (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 360 of file coder_array.h.

```
360
361     return i_ < _other.i_;
362 }
```

6.9.3.14 operator<=()

```
template<typename T >
bool coder::const_array_iterator< T >::operator<= (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 366 of file coder_array.h.

```
366
367     return i_ <= _other.i_;
368 }
```

6.9.3.15 operator=()

```
template<typename T >
const_array_iterator<T>& coder::const_array_iterator< T >::operator= (
    const const_array_iterator< T > & _other ) [inline]
```

Definition at line 350 of file coder_array.h.

```
350
351     this->i_ = _other.i_;
352     return *this;
353 }
```

6.9.3.16 operator==()

```
template<typename T >
bool coder::const_array_iterator< T >::operator== (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 354 of file coder_array.h.

```
354
355     return i_ == _other.i_;
356 }
```

6.9.3.17 operator>()

```
template<typename T >
bool coder::const_array_iterator< T >::operator> (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 363 of file coder_array.h.

```
363
364     return i_ > _other.i_;
365 }
```

6.9.3.18 operator>=()

```
template<typename T >
bool coder::const_array_iterator< T >::operator>= (
    const const_array_iterator< T > & _other ) const [inline]
```

Definition at line 369 of file coder_array.h.

```
369
370     return i_ >= _other.i_;
371 }
```

6.9.3.19 `operator[]()`

```
template<typename T >
const T::value_type& coder::const_array_iterator< T >::operator[] (
```

typename T::size_type _di) const [inline]

Definition at line 329 of file `coder_array.h`.

```
329
330     return (*arr_) [i_ + _di];
331 }
```

6.9.4 Member Data Documentation

6.9.4.1 `arr_`

```
template<typename T >
const T* coder::const_array_iterator< T >::arr_ [private]
```

Definition at line 402 of file `coder_array.h`.

6.9.4.2 `i_`

```
template<typename T >
T::size_type coder::const_array_iterator< T >::i_ [private]
```

Definition at line 403 of file `coder_array.h`.

6.9.4.3 `n_`

```
template<typename T >
T::size_type coder::const_array_iterator< T >::n_ [private]
```

Definition at line 403 of file `coder_array.h`.

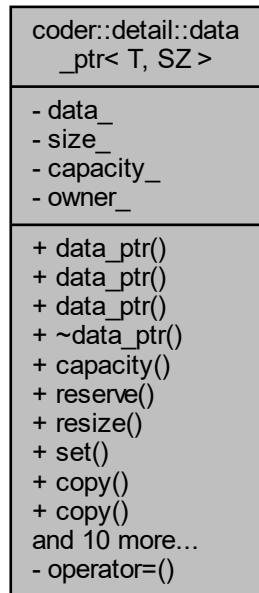
The documentation for this class was generated from the following file:

- `codegen/lib(Func_fovPathPlanning/coder_array.h`

6.10 coder::detail::data_ptr< T, SZ > Class Template Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::data_ptr< T, SZ >:



Public Types

- `typedef T value_type`
- `typedef SZ size_type`

Public Member Functions

- `data_ptr ()`
- `data_ptr (T *_data, SZ _sz)`
- `data_ptr (const data_ptr &_other)`
- `~data_ptr ()`
- `SZ capacity () const`
- `void reserve (SZ _n)`
- `void resize (SZ _n)`
- `void set (T *_data, const SZ _sz)`
- `void copy (const T *_data, SZ _size)`
- `void copy (const data_ptr< T, SZ > &_other)`
- `operator T* ()`
- `operator const T * () const`
- `T & operator[] (SZ _index)`

- const T & [operator\[\]](#) (SZ _index) const
- T * [operator->](#) ()
- const T * [operator->](#) () const
- bool [is_null](#) () const
- void [clear](#) ()
- bool [is_owner](#) () const
- void [set_owner](#) (bool _b)

Private Member Functions

- void [operator=](#) (const [data_ptr](#)< T, SZ > &_other)

Private Attributes

- T * [data_](#)
- SZ [size_](#)
- SZ [capacity_](#)
- bool [owner_](#)

6.10.1 Detailed Description

```
template<typename T, typename SZ>
class coder::detail::data_ptr< T, SZ >
```

Definition at line 66 of file coder_array.h.

6.10.2 Member Typedef Documentation

6.10.2.1 size_type

```
template<typename T , typename SZ >
typedef SZ coder::detail::data\_ptr< T, SZ >::size\_type
```

Definition at line 69 of file coder_array.h.

6.10.2.2 value_type

```
template<typename T , typename SZ >
typedef T coder::detail::data\_ptr< T, SZ >::value\_type
```

Definition at line 68 of file coder_array.h.

6.10.3 Constructor & Destructor Documentation

6.10.3.1 data_ptr() [1/3]

```
template<typename T , typename SZ >
coder::detail::data_ptr< T, SZ >::data_ptr ( ) [inline]
```

Definition at line 71 of file coder_array.h.

```
72     : data_(NULL)
73     , size_(0)
74     , capacity_(0)
75     , owner_(false) {
76 }
```

6.10.3.2 data_ptr() [2/3]

```
template<typename T , typename SZ >
coder::detail::data_ptr< T, SZ >::data_ptr (
    T * _data,
    SZ _sz ) [inline]
```

Definition at line 77 of file coder_array.h.

```
78     : data_(_data)
79     , size_(_sz)
80     , capacity_(_sz)
81     , owner_(false) {
82 }
```

6.10.3.3 data_ptr() [3/3]

```
template<typename T , typename SZ >
coder::detail::data_ptr< T, SZ >::data_ptr (
    const data_ptr< T, SZ > & _other ) [inline]
```

Definition at line 84 of file coder_array.h.

```
85     : data_(_other.owner_ ? NULL : _other.data_)
86     , size_(_other.owner_ ? 0 : _other.size_)
87     , capacity_(_other.owner_ ? 0 : _other.capacity_)
88     , owner_(_other.owner_) {
89     if (owner_) {
90         resize(_other.size_);
91         std::copy(_other.data_, _other.data_ + size_, data_);
92     }
93 }
```

6.10.3.4 ~data_ptr()

```
template<typename T , typename SZ >
coder::detail::data_ptr< T, SZ >::~data_ptr ( ) [inline]
```

Definition at line 95 of file coder_array.h.

```
95         {
96             if (owner_) {
97                 CODER_DELETE(data_);
98             }
99         }
```

6.10.4 Member Function Documentation

6.10.4.1 capacity()

```
template<typename T , typename SZ >
SZ coder::detail::data_ptr< T, SZ >::capacity ( ) const [inline]
```

Definition at line 100 of file coder_array.h.

```
100         {
101             return capacity_;
102         }
```

6.10.4.2 clear()

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::clear ( ) [inline]
```

Definition at line 181 of file coder_array.h.

```
181         {
182             if (owner_) {
183                 CODER_DELETE(data_);
184             }
185             data_ = NULL;
186             size_ = 0;
187             capacity_ = 0;
188             owner_ = false;
189         }
```

6.10.4.3 copy() [1/2]

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::copy (
    const data_ptr< T, SZ > & _other ) [inline]
```

Definition at line 150 of file coder_array.h.

```
150         {
151             copy(_other.data_, _other.size_);
152         }
```

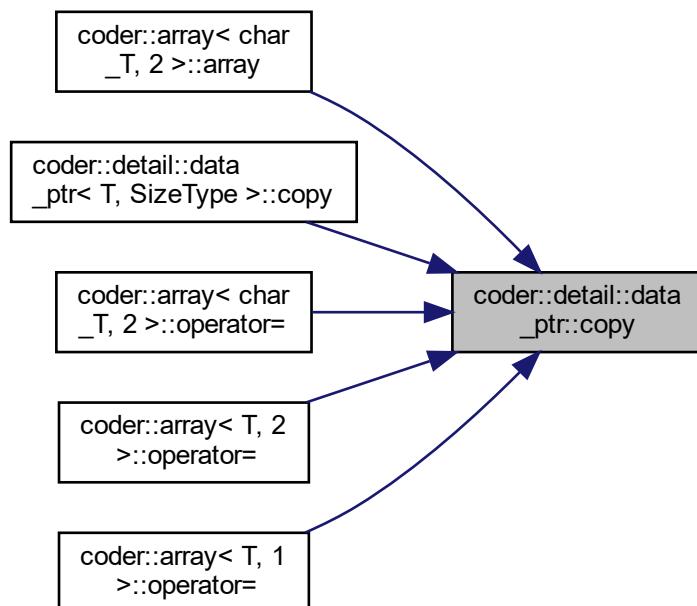
6.10.4.4 copy() [2/2]

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::copy (
    const T * _data,
    SZ _size ) [inline]
```

Definition at line 135 of file coder_array.h.

```
135
136     if (data_ == _data) {
137         size_ = _size;
138         return;
139     }
140     if (owner_) {
141         CODER_DELETE(data_);
142     }
143     data_ = CODER_NEW(T, _size);
144     owner_ = true;
145     size_ = _size;
146     capacity_ = size_;
147     std::copy(_data, _data + _size, data_);
148 }
```

Here is the caller graph for this function:



6.10.4.5 is_null()

```
template<typename T , typename SZ >
bool coder::detail::data_ptr< T, SZ >::is_null ( ) const [inline]
```

Definition at line 177 of file coder_array.h.

```
177
178     {
179     return data_ == NULL;
180 }
```

6.10.4.6 is_owner()

```
template<typename T , typename SZ >
bool coder::detail::data_ptr< T, SZ >::is_owner ( ) const [inline]
```

Definition at line 191 of file coder_array.h.

```
191      {
192         return owner_;
193     }
```

6.10.4.7 operator const T *()

```
template<typename T , typename SZ >
coder::detail::data_ptr< T, SZ >::operator const T * ( ) const [inline]
```

Definition at line 158 of file coder_array.h.

```
158      {
159         return &data_[0];
160     }
```

6.10.4.8 operator T*()

```
template<typename T , typename SZ >
coder::detail::data_ptr< T, SZ >::operator T* ( ) [inline]
```

Definition at line 154 of file coder_array.h.

```
154      {
155         return &data_[0];
156     }
```

6.10.4.9 operator->() [1/2]

```
template<typename T , typename SZ >
T* coder::detail::data_ptr< T, SZ >::operator-> ( ) [inline]
```

Definition at line 169 of file coder_array.h.

```
169      {
170         return data_;
171     }
```

6.10.4.10 operator->() [2/2]

```
template<typename T , typename SZ >
const T* coder::detail::data_ptr< T, SZ >::operator-> ( ) const [inline]
```

Definition at line 173 of file coder_array.h.

```
173      {
174         return data_;
175     }
```

6.10.4.11 operator=()

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::operator= (
    const data_ptr< T, SZ > & _other ) [private]
```

6.10.4.12 operator[]() [1/2]

```
template<typename T , typename SZ >
T& coder::detail::data_ptr< T, SZ >::operator[] (
    SZ _index ) [inline]
```

Definition at line 162 of file coder_array.h.

```
162
163     return data_[_index];
164 }
```

6.10.4.13 operator[]() [2/2]

```
template<typename T , typename SZ >
const T& coder::detail::data_ptr< T, SZ >::operator[] (
    SZ _index ) const [inline]
```

Definition at line 165 of file coder_array.h.

```
165
166     return data_[_index];
167 }
```

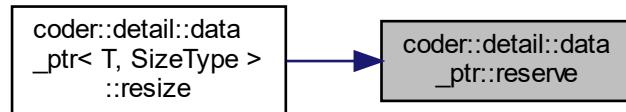
6.10.4.14 reserve()

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::reserve (
    SZ _n ) [inline]
```

Definition at line 103 of file coder_array.h.

```
103
104     if (_n > capacity_) {
105         T* new_data = CODER_NEW(T, _n);
106         std::copy(data_, data_ + size_, new_data);
107         if (owner_) {
108             CODER_DELETE(data_);
109         }
110         data_ = new_data;
111         capacity_ = _n;
112         owner_ = true;
113     }
114 }
```

Here is the caller graph for this function:



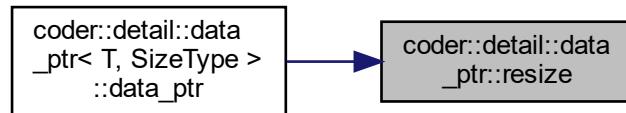
6.10.4.15 resize()

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::resize (
    SZ _n ) [inline]
```

Definition at line 115 of file coder_array.h.

```
115     {
116         reserve(_n);
117         size_ = _n;
118     }
```

Here is the caller graph for this function:



6.10.4.16 set()

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::set (
    T * _data,
    const SZ _sz ) [inline]
```

Definition at line 125 of file coder_array.h.

```
125     {
126         if (owner_) {
127             CODER_DELETE(data_);
128         }
129         data_ = _data;
130         size_ = _sz;
131         owner_ = false;
132         capacity_ = size_;
133     }
```

6.10.4.17 `set_owner()`

```
template<typename T , typename SZ >
void coder::detail::data_ptr< T, SZ >::set_owner (
    bool _b ) [inline]
```

Definition at line 195 of file `coder_array.h`.

```
195         {
196     owner_ = _b;
197 }
```

6.10.5 Member Data Documentation

6.10.5.1 `capacity_`

```
template<typename T , typename SZ >
SZ coder::detail::data_ptr< T, SZ >::capacity_ [private]
```

Definition at line 202 of file `coder_array.h`.

6.10.5.2 `data_`

```
template<typename T , typename SZ >
T* coder::detail::data_ptr< T, SZ >::data_ [private]
```

Definition at line 200 of file `coder_array.h`.

6.10.5.3 `owner_`

```
template<typename T , typename SZ >
bool coder::detail::data_ptr< T, SZ >::owner_ [private]
```

Definition at line 203 of file `coder_array.h`.

6.10.5.4 `size_`

```
template<typename T , typename SZ >
SZ coder::detail::data_ptr< T, SZ >::size_ [private]
```

Definition at line 201 of file `coder_array.h`.

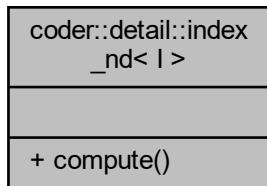
The documentation for this class was generated from the following file:

- `codegen/lib(Func_fovPathPlanning/coder_array.h`

6.11 coder::detail::index_nd< I > Class Template Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::index_nd< I >:



Static Public Member Functions

- template<typename SZ >
static SZ **compute** (const SZ _size[], const SZ _indices[])

6.11.1 Detailed Description

```
template<int I>
class coder::detail::index_nd< I >
```

Definition at line 428 of file coder_array.h.

6.11.2 Member Function Documentation

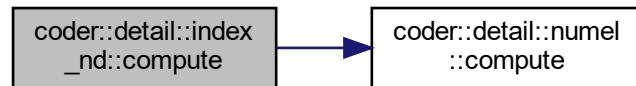
6.11.2.1 compute()

```
template<int I>
template<typename SZ >
static SZ coder::detail::index_nd< I >::compute (
    const SZ _size[],
    const SZ _indices[] ) [inline], [static]
```

Definition at line 431 of file coder_array.h.

```
431
432     const SZ weight = numel<I - 1>::compute(_size);
433     return weight * _indices[I - 1] + index_nd<I - 1>::compute(_size, _indices);
434 }
```

Here is the call graph for this function:



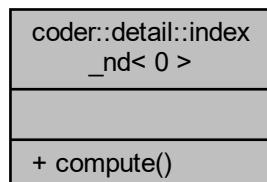
The documentation for this class was generated from the following file:

- codegen/lib(Func_fovPathPlanning/coder_array.h)

6.12 coder::detail::index_nd< 0 > Class Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::index_nd< 0 >:



Static Public Member Functions

- template<typename SZ >
static SZ [compute](#) (SZ[], SZ[])

6.12.1 Detailed Description

Definition at line 438 of file [coder_array.h](#).

6.12.2 Member Function Documentation

6.12.2.1 compute()

```
template<typename SZ >
static SZ coder::detail::index_nd< 0 >::compute (
    SZ [ ],
    SZ [ ] )  [inline], [static]

Definition at line 441 of file coder_array.h.
441
442     return 0;
443 }
```

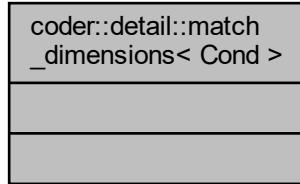
The documentation for this class was generated from the following file:

- codegen/lib/Func_fovPathPlanning/coder_array.h

6.13 coder::detail::match_dimensions< Cond > Struct Template Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::match_dimensions< Cond >:



6.13.1 Detailed Description

```
template<bool Cond>
struct coder::detail::match_dimensions< Cond >
```

Definition at line 447 of file coder_array.h.

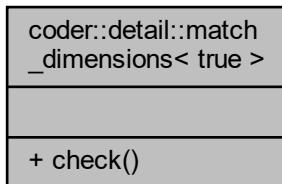
The documentation for this struct was generated from the following file:

- codegen/lib/Func_fovPathPlanning/coder_array.h

6.14 coder::detail::match_dimensions< true > Struct Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::match_dimensions< true >:



Static Public Member Functions

- static void [check \(\)](#)

6.14.1 Detailed Description

Definition at line 450 of file `coder_array.h`.

6.14.2 Member Function Documentation

6.14.2.1 [check\(\)](#)

```
static void coder::detail::match_dimensions< true >::check ( ) [inline], [static]
```

Definition at line 451 of file `coder_array.h`.

```
451 {  
452 }
```

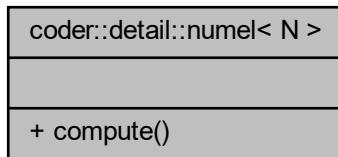
The documentation for this struct was generated from the following file:

- `codegen/lib(Func_fovPathPlanning/coder_array.h`

6.15 coder::detail::numel< N > Class Template Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::numel< N >:



Static Public Member Functions

- template<typename SZ>
static SZ [compute](#) (SZ _size[])

6.15.1 Detailed Description

```
template<int N>
class coder::detail::numel< N >
```

Definition at line 410 of file `coder_array.h`.

6.15.2 Member Function Documentation

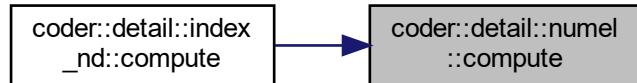
6.15.2.1 [compute\(\)](#)

```
template<int N>
template<typename SZ >
static SZ coder::detail::numel< N >::compute (
    SZ _size[] ) [inline], [static]
```

Definition at line 413 of file `coder_array.h`.

```
413
414     {
415     return _size[N - 1] * numel<N - 1>::compute(_size);
}
```

Here is the caller graph for this function:



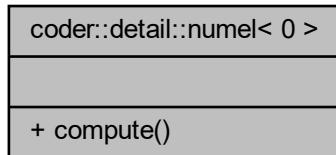
The documentation for this class was generated from the following file:

- codegen/lib/FunPathPlanning/coder_array.h

6.16 coder::detail::numel< 0 > Class Reference

```
#include <coder_array.h>
```

Collaboration diagram for coder::detail::numel< 0 >:



Static Public Member Functions

- template<typename SZ>
static SZ [compute](#)(SZ[])

6.16.1 Detailed Description

Definition at line 418 of file [coder_array.h](#).

6.16.2 Member Function Documentation

6.16.2.1 compute()

```
template<typename SZ >
static SZ coder::detail::numel< 0 >::compute (
    SZ [ ] ) [inline], [static]
```

Definition at line 421 of file coder_array.h.

```
421             {
422         return 1;
423     }
```

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

- codegen/lib(Func_fovPathPlanning/coder_array.h)

6.17 struct_T Struct Reference

```
#include <Func_fovPathPlanning_types.h>
```

Collaboration diagram for struct_T:



Public Attributes

- double **fov**
- double **dFOV**
- double **radius**
- double **dvec**
- double **scolor**
- double **dc**
- double **Rmin**
- double **size** [2]
- double **cs**
- double **mindis**
- double **H**
- double **W**

6.17.1 Detailed Description

Definition at line 36 of file Func_fovPathPlanning_types.h.

6.17.2 Member Data Documentation

6.17.2.1 cs

```
double struct_T::cs
```

Definition at line 46 of file Func_fovPathPlanning_types.h.

6.17.2.2 dc

```
double struct_T::dc
```

Definition at line 43 of file Func_fovPathPlanning_types.h.

6.17.2.3 dFOV

```
double struct_T::dFOV
```

Definition at line 39 of file Func_fovPathPlanning_types.h.

6.17.2.4 dvec

```
double struct_T::dvec
```

Definition at line 41 of file Func_fovPathPlanning_types.h.

6.17.2.5 fov

```
double struct_T::fov
```

Definition at line 38 of file Func_fovPathPlanning_types.h.

6.17.2.6 H

```
double struct_T::H
```

Definition at line 48 of file Func_fovPathPlanning_types.h.

6.17.2.7 mindis

```
double struct_T::mindis
```

Definition at line 47 of file Func_fovPathPlanning_types.h.

6.17.2.8 radius

```
double struct_T::radius
```

Definition at line 40 of file Func_fovPathPlanning_types.h.

6.17.2.9 Rmin

```
double struct_T::Rmin
```

Definition at line 44 of file Func_fovPathPlanning_types.h.

6.17.2.10 scolor

```
double struct_T::scolor
```

Definition at line 42 of file Func_fovPathPlanning_types.h.

6.17.2.11 size

```
double struct_T::size[2]
```

Definition at line 45 of file Func_fovPathPlanning_types.h.

6.17.2.12 W

```
double struct_T::W
```

Definition at line 49 of file Func_fovPathPlanning_types.h.

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

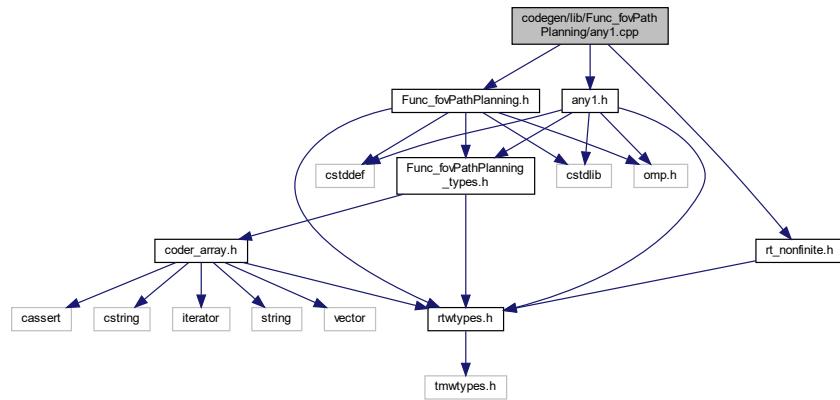
- codegen/lib(Func_fovPathPlanning)[Func_fovPathPlanning_types.h](#)

Chapter 7

File Documentation

7.1 codegen/lib(Func_fovPathPlanning/any1.cpp File Reference

```
#include "any1.h"
#include "Func_fovPathPlanning.h"
#include "rt_nonfinite.h"
Include dependency graph for any1.cpp:
```



Functions

- void `any` (const double x_data[], const int x_size[2], boolean_T y[3])

7.1.1 Function Documentation

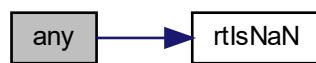
7.1.1.1 any()

```
void any (
    const double x_data[],
    const int x_size[2],
    boolean_T y{3} )
```

Definition at line 18 of file any1.cpp.

```
19 {
20     int ix;
21     boolean_T exitgl;
22     int a_tmp;
23     int a;
24     y[0] = false;
25     y[1] = false;
26     y[2] = false;
27     ix = 0;
28     exitgl = false;
29     while ((!exitgl) && (ix + 1 <= x_size[0])) {
30         if ((x_data[ix] == 0.0) || rtIsNaN(x_data[ix])) {
31             ix++;
32         } else {
33             y[0] = true;
34             exitgl = true;
35         }
36     }
37
38     a_tmp = x_size[0] + x_size[0];
39     ix = x_size[0];
40     exitgl = false;
41     while ((!exitgl) && (ix + 1 <= a_tmp)) {
42         if ((x_data[ix] == 0.0) || rtIsNaN(x_data[ix])) {
43             ix++;
44         } else {
45             y[1] = true;
46             exitgl = true;
47         }
48     }
49
50     a = a_tmp + x_size[0];
51     ix = a_tmp;
52     exitgl = false;
53     while ((!exitgl) && (ix + 1 <= a)) {
54         if ((x_data[ix] == 0.0) || rtIsNaN(x_data[ix])) {
55             ix++;
56         } else {
57             y[2] = true;
58             exitgl = true;
59         }
60     }
61 }
```

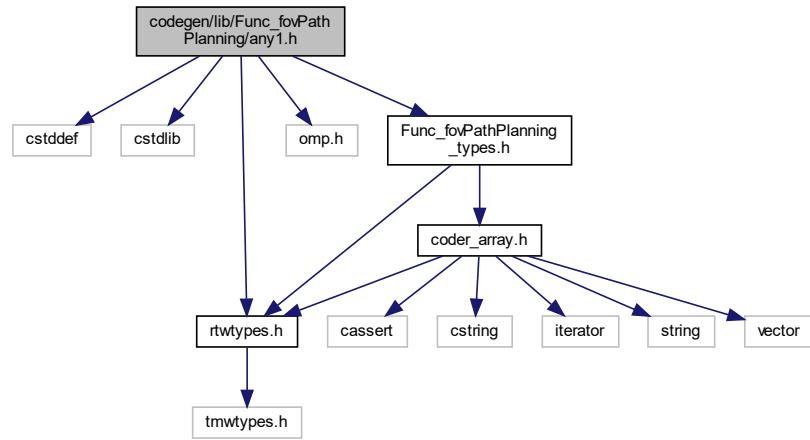
Here is the call graph for this function:



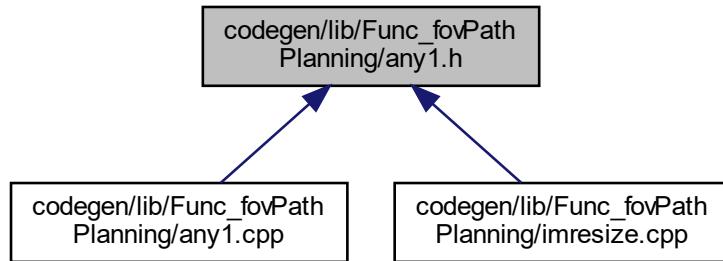
7.2 codegen/lib/Func_fovPathPlanning/any1.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
```

```
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for any1.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void any (const double x_data[], const int x_size[2], boolean_T y[3])`

7.2.1 Macro Definition Documentation

7.2.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file any1.h.

7.2.2 Function Documentation

7.2.2.1 any()

```
void any (
    const double x_data[],
    const int x_size[2],
    boolean_T y[3] )
```

Definition at line 18 of file any1.cpp.

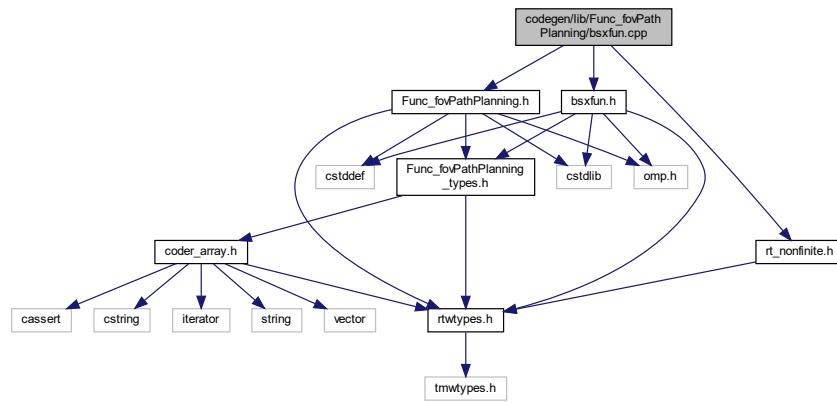
```
19 {
20     int ix;
21     boolean_T exitgl;
22     int a_tmp;
23     int a;
24     y[0] = false;
25     y[1] = false;
26     y[2] = false;
27     ix = 0;
28     exitgl = false;
29     while ((!exitgl) && (ix + 1 <= x_size[0])) {
30         if ((x_data[ix] == 0.0) || rtIsNaN(x_data[ix])) {
31             ix++;
32         } else {
33             y[0] = true;
34             exitgl = true;
35         }
36     }
37
38     a_tmp = x_size[0] + x_size[0];
39     ix = x_size[0];
40     exitgl = false;
41     while ((!exitgl) && (ix + 1 <= a_tmp)) {
42         if ((x_data[ix] == 0.0) || rtIsNaN(x_data[ix])) {
43             ix++;
44         } else {
45             y[1] = true;
46             exitgl = true;
47         }
48     }
49
50     a = a_tmp + x_size[0];
51     ix = a_tmp;
52     exitgl = false;
53     while ((!exitgl) && (ix + 1 <= a)) {
54         if ((x_data[ix] == 0.0) || rtIsNaN(x_data[ix])) {
55             ix++;
56         } else {
57             y[2] = true;
58             exitgl = true;
59         }
60     }
61 }
```

Here is the call graph for this function:



7.3 codegen/lib/Func_fovPathPlanning/bsxfun.cpp File Reference

```
#include "bsxfun.h"
#include "Func_fovPathPlanning.h"
#include "rt_nonfinite.h"
Include dependency graph for bsxfun.cpp:
```



Functions

- void `b_bsxfun` (const double a_data[], const int a_size[1], const double b_data[], const int b_size[2], double c_data[], int c_size[2])
- void `bsxfun` (const int a_data[], const int a_size[1], int c_data[], int c_size[2])
- void `c_bsxfun` (const double a_data[], const int a_size[2], const double b_data[], const int b_size[1], double c_data[], int c_size[2])

7.3.1 Function Documentation

7.3.1.1 `b_bsxfun()`

```
void b_bsxfun (
    const double a_data[],
    const int a_size[1],
    const double b_data[],
    const int b_size[2],
    double c_data[],
    int c_size[2] )
```

Definition at line 18 of file `bsxfun.cpp`.

```
20 {
21     int acoef;
22     int bcoef;
23     int i;
24     if (b_size[0] == 1) {
25         c_size[0] = static_cast<signed char>(a_size[0]);
26     } else if (a_size[0] == 1) {
27         c_size[0] = static_cast<signed char>(b_size[0]);
```

```

28 } else if (a_size[0] == b_size[0]) {
29     c_size[0] = static_cast<signed char>(a_size[0]);
30 } else if (b_size[0] < a_size[0]) {
31     c_size[0] = static_cast<signed char>(b_size[0]);
32 } else {
33     c_size[0] = static_cast<signed char>(a_size[0]);
34 }
35
36 c_size[1] = 3;
37 acoef = (a_size[0] != 1);
38 bcoef = (b_size[0] != 1);
39 i = c_size[0] - 1;
40 for (int k = 0; k < 3; k++) {
41     for (int b_k = 0; b_k <= i; b_k++) {
42         c_data[b_k + c_size[0] * k] = a_data[acoef * b_k] - b_data[bcoef * b_k +
43             b_size[0] * k];
44     }
45 }
46 }
```

7.3.1.2 bsxfun()

```

void bsxfun (
    const int a_data[],
    const int a_size[1],
    int c_data[],
    int c_size[2] )
```

Definition at line 48 of file bsxfun.cpp.

```

49 {
50     int acoef;
51     int i;
52     c_size[0] = static_cast<signed char>(a_size[0]);
53     c_size[1] = 3;
54     acoef = (a_size[0] != 1);
55     i = c_size[0] - 1;
56     for (int k = 0; k < 3; k++) {
57         for (int b_k = 0; b_k <= i; b_k++) {
58             c_data[b_k + c_size[0] * k] = a_data[acoef * b_k] + k;
59         }
60     }
61 }
```

7.3.1.3 c_bsxfun()

```

void c_bsxfun (
    const double a_data[],
    const int a_size[2],
    const double b_data[],
    const int b_size[1],
    double c_data[],
    int c_size[2] )
```

Definition at line 63 of file bsxfun.cpp.

```

65 {
66     int acoef;
67     int bcoef;
68     int i;
69     if (b_size[0] == 1) {
70         c_size[0] = static_cast<signed char>(a_size[0]);
71     } else if (a_size[0] == 1) {
72         c_size[0] = static_cast<signed char>(b_size[0]);
73     } else if (a_size[0] == b_size[0]) {
74         c_size[0] = static_cast<signed char>(a_size[0]);
75     } else if (b_size[0] < a_size[0]) {
```

```

76     c_size[0] = static_cast<signed char>(b_size[0]);
77 } else {
78     c_size[0] = static_cast<signed char>(a_size[0]);
79 }
80
81 c_size[1] = 3;
82 acoef = (a_size[0] != 1);
83 bcoef = (b_size[0] != 1);
84 i = c_size[0] - 1;
85 for (int k = 0; k < 3; k++) {
86     for (int b_k = 0; b_k <= i; b_k++) {
87         c_data[b_k + c_size[0] * k] = a_data[acoef * b_k + a_size[0] * k] /
88             b_data[bcoef * b_k];
89     }
90 }
91 }

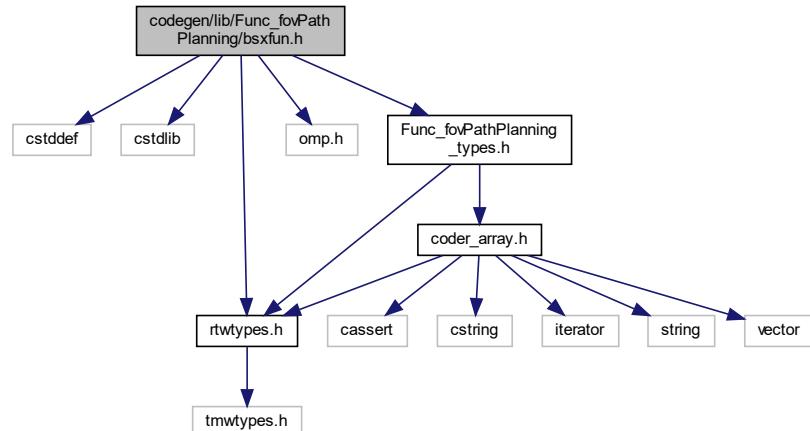
```

7.4 codegen/lib(Func_fovPathPlanning/bsxfun.h File Reference

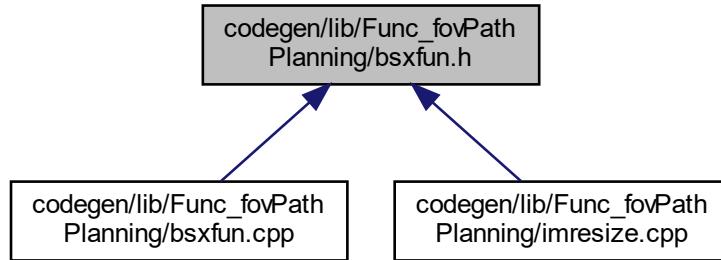
```

#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for bsxfun.h:

```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void b_bsxfun (const double a_data[], const int a_size[1], const double b_data[], const int b_size[2], double c_data[], int c_size[2])`
- `void bsxfun (const int a_data[], const int a_size[1], int c_data[], int c_size[2])`
- `void c_bsxfun (const double a_data[], const int a_size[2], const double b_data[], const int b_size[1], double c_data[], int c_size[2])`

7.4.1 Macro Definition Documentation

7.4.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file bsxfun.h.

7.4.2 Function Documentation

7.4.2.1 b_bsxfun()

```
void b_bsxfun (
    const double a_data[],
    const int a_size[1],
    const double b_data[],
    const int b_size[2],
    double c_data[],
    int c_size[2] )
```

Definition at line 18 of file bsxfun.cpp.

```
20 {
21     int acoef;
22     int bcoef;
23     int i;
24     if (b_size[0] == 1) {
25         c_size[0] = static_cast<signed char>(a_size[0]);
26     } else if (a_size[0] == 1) {
27         c_size[0] = static_cast<signed char>(b_size[0]);
28     } else if (a_size[0] == b_size[0]) {
29         c_size[0] = static_cast<signed char>(a_size[0]);
30     } else if (b_size[0] < a_size[0]) {
31         c_size[0] = static_cast<signed char>(b_size[0]);
32     } else {
33         c_size[0] = static_cast<signed char>(a_size[0]);
34     }
35
36     c_size[1] = 3;
37     acoef = (a_size[0] != 1);
38     bcoef = (b_size[0] != 1);
39     i = c_size[0] - 1;
40     for (int k = 0; k < 3; k++) {
41         for (int b_k = 0; b_k <= i; b_k++) {
42             c_data[b_k + c_size[0] * k] = a_data[acoef * b_k] - b_data[bcoef * b_k +
43                 b_size[0] * k];
44         }
45     }
46 }
```

7.4.2.2 bsxfun()

```
void bsxfun (
    const int a_data[],
    const int a_size[1],
    int c_data[],
    int c_size[2] )
```

Definition at line 48 of file bsxfun.cpp.

```
49 {
50     int acoef;
51     int i;
52     c_size[0] = static_cast<signed char>(a_size[0]);
53     c_size[1] = 3;
54     acoef = (a_size[0] != 1);
55     i = c_size[0] - 1;
56     for (int k = 0; k < 3; k++) {
57         for (int b_k = 0; b_k <= i; b_k++) {
58             c_data[b_k + c_size[0] * k] = a_data[acoef * b_k] + k;
59         }
60     }
61 }
```

7.4.2.3 c_bsxfun()

```
void c_bsxfun (
    const double a_data[],
    const int a_size[2],
    const double b_data[],
    const int b_size[1],
    double c_data[],
    int c_size[2] )
```

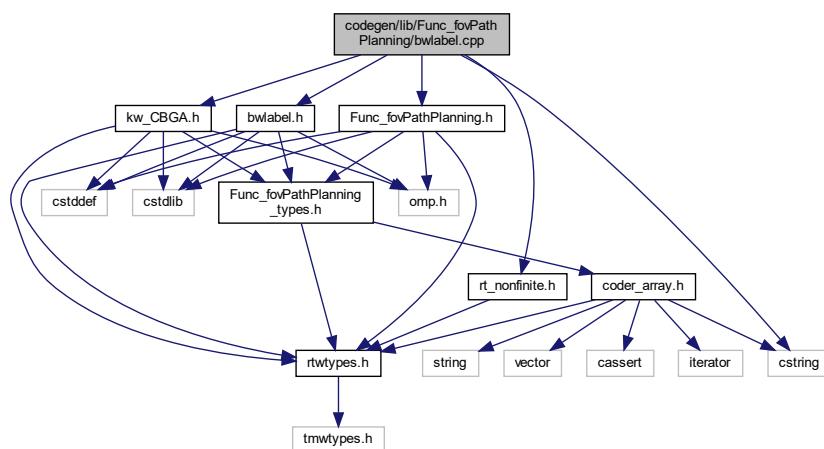
Definition at line 63 of file bsxfun.cpp.

```
65 {
66     int acoef;
67     int bcoef;
68     int i;
69     if (b_size[0] == 1) {
70         c_size[0] = static_cast<signed char>(a_size[0]);
71     } else if (a_size[0] == 1) {
72         c_size[0] = static_cast<signed char>(b_size[0]);
73     } else if (a_size[0] == b_size[0]) {
74         c_size[0] = static_cast<signed char>(a_size[0]);
75     } else if (b_size[0] < a_size[0]) {
76         c_size[0] = static_cast<signed char>(b_size[0]);
77     } else {
78         c_size[0] = static_cast<signed char>(a_size[0]);
79     }
80
81     c_size[1] = 3;
82     acoef = (a_size[0] != 1);
83     bcoef = (b_size[0] != 1);
84     i = c_size[0] - 1;
85     for (int k = 0; k < 3; k++) {
86         for (int b_k = 0; b_k <= i; b_k++) {
87             c_data[b_k + c_size[0] * k] = a_data[acoef * b_k + a_size[0] * k] /
88             b_data[bcoef * b_k];
89     }
90 }
```

7.5 codegen/lib/Func_fovPathPlanning/bwlabel.cpp File Reference

```
#include "bwlabel.h"
#include "Func_fovPathPlanning.h"
#include "kw_CBGA.h"
#include "rt_nonfinite.h"
#include <cstring>
```

Include dependency graph for bwlabel.cpp:



Functions

- void **b_bwlabel** (const unsigned long long varargin_1[6000000], double L[6000000])
- void **bwlabel** (const unsigned long long varargin_1[2542], double L[2542])
- void **c_bwlabel** (const unsigned long long varargin_1_data[], const int varargin_1_size[2], double L_data[], int L_size[2])

7.5.1 Function Documentation

7.5.1.1 b_bwlabel()

```
void b_bwlabel (
    const unsigned long long varargin_1[6000000],
    double L[6000000] )
```

Definition at line 20 of file bwlabel.cpp.

```
21 {
22     double numComponents;
23     int firstRunOnPreviousColumn;
24     int numRuns;
25     static boolean_T im[6000000];
26     int lastRunOnPreviousColumn;
27     coder::array<short, 1U> startRow;
28     coder::array<short, 1U> endRow;
29     int k;
30     coder::array<short, 1U> startCol;
31     coder::array<int, 1U> labelForEachRun;
32     coder::array<int, 1U> labelsRenumbered;
33     numComponents = 0.0;
34     for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < 6000000;
35         firstRunOnPreviousColumn++) {
36         im[firstRunOnPreviousColumn] = (varargin_1[firstRunOnPreviousColumn] != 0ULL);
37         L[firstRunOnPreviousColumn] = 0.0;
38     }
39
40     numRuns = 0;
41     for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 2000;
42         lastRunOnPreviousColumn++) {
43         if (im[3000 * lastRunOnPreviousColumn]) {
44             numRuns++;
45         }
46
47         for (k = 0; k < 2999; k++) {
48             firstRunOnPreviousColumn = k + 3000 * lastRunOnPreviousColumn;
49             if (im[firstRunOnPreviousColumn + 1] && (!im[firstRunOnPreviousColumn])) {
50                 numRuns++;
51             }
52         }
53     }
54
55     if (numRuns != 0) {
56         int runCounter;
57         int row;
58         int firstRunOnThisColumn;
59         startRow.set_size(numRuns);
60         endRow.set_size(numRuns);
61         startCol.set_size(numRuns);
62         runCounter = 0;
63         for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 2000;
64             lastRunOnPreviousColumn++) {
65             row = 1;
66             while (row <= 3000) {
67                 while ((row <= 3000) && (!im[(row + 3000 * lastRunOnPreviousColumn) - 1]))
68                 {
69                     row++;
70                 }
71
72                 if ((row <= 3000) && im[(row + 3000 * lastRunOnPreviousColumn) - 1]) {
73                     startCol[runCounter] = static_cast<short>(lastRunOnPreviousColumn + 1);
74                     startRow[runCounter] = static_cast<short>(row);
75                     while ((row <= 3000) && im[(row + 3000 * lastRunOnPreviousColumn) - 1])
```

```

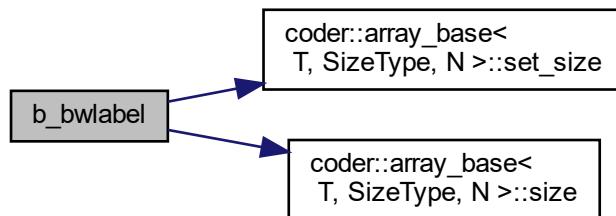
76         {
77             row++;
78         }
79
80         endRow[runCounter] = static_cast<short>(row - 1);
81         runCounter++;
82     }
83 }
84
85
86 labelForEachRun.set_size(numRuns);
87 for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < numRuns;
88       firstRunOnPreviousColumn++) {
89     labelForEachRun[firstRunOnPreviousColumn] = 0;
90 }
91
92 k = 0;
93 runCounter = 1;
94 row = 1;
95 firstRunOnPreviousColumn = -1;
96 lastRunOnPreviousColumn = -1;
97 firstRunOnThisColumn = 0;
98 while (k + 1 <= numRuns) {
99     if (startCol[k] == runCounter + 1) {
100        firstRunOnPreviousColumn = firstRunOnThisColumn + 1;
101        firstRunOnThisColumn = k;
102        lastRunOnPreviousColumn = k;
103        runCounter = startCol[k];
104    } else {
105        if (startCol[k] > runCounter + 1) {
106            firstRunOnPreviousColumn = -1;
107            lastRunOnPreviousColumn = -1;
108            firstRunOnThisColumn = k;
109            runCounter = startCol[k];
110        }
111    }
112
113    if (firstRunOnPreviousColumn >= 0) {
114        for (int p = firstRunOnPreviousColumn - 1; p < lastRunOnPreviousColumn;
115              p++) {
116            if ((endRow[k] >= startRow[p] - 1) && (startRow[k] <= endRow[p] + 1))
117            {
118                if (labelForEachRun[k] == 0) {
119                    labelForEachRun[k] = labelForEachRun[p];
120                    row++;
121                } else {
122                    if (labelForEachRun[k] != labelForEachRun[p]) {
123                        int root_k;
124                        int root_p;
125                        for (root_k = k; root_k + 1 != labelForEachRun[root_k]; root_k =
126                            labelForEachRun[root_k] - 1) {
127                            labelForEachRun[root_k] =
128                                labelForEachRun[labelForEachRun[root_k] - 1];
129                        }
130
131                        for (root_p = p; root_p + 1 != labelForEachRun[root_p]; root_p =
132                            labelForEachRun[root_p] - 1) {
133                            labelForEachRun[root_p] =
134                                labelForEachRun[labelForEachRun[root_p] - 1];
135                        }
136
137                        if (root_k + 1 != root_p + 1) {
138                            if (root_p + 1 < root_k + 1) {
139                                labelForEachRun[root_k] = root_p + 1;
140                                labelForEachRun[k] = root_p + 1;
141                            } else {
142                                labelForEachRun[root_p] = root_k + 1;
143                                labelForEachRun[p] = root_k + 1;
144                            }
145                        }
146                    }
147                }
148            }
149        }
150    }
151
152    if (labelForEachRun[k] == 0) {
153        labelForEachRun[k] = row;
154        row++;
155    }
156
157    k++;
158 }
159
160 labelsRenumbered.set_size(labelForEachRun.size(0));
161 for (k = 0; k < numRuns; k++) {
162     if (labelForEachRun[k] == k + 1) {

```

```

163     numComponents++;
164     labelsRenumbered[k] = static_cast<int>(numComponents);
165 }
166
167 labelsRenumbered[k] = labelsRenumbered[labelForEachRun[k] - 1];
168 firstRunOnPreviousColumn = startRow[k];
169 runCounter = endRow[k];
170 for (row = firstRunOnPreviousColumn; row <= runCounter; row++) {
171     L[(row + 3000 * (startCol[k] - 1)) - 1] = labelsRenumbered[k];
172 }
173 }
174 }
175 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.1.2 bwlabel()

```

void bwlabel (
    const unsigned long long varargin_1[2542],
    double L[2542] )
```

Definition at line 177 of file bwlabel.cpp.

```

178 {
179     double numComponents;
180     int firstRunOnPreviousColumn;
181     int numRuns;
182     boolean_T im[2542];
183     int lastRunOnPreviousColumn;
184     coder::array<signed char, 1U> startRow;
185     coder::array<signed char, 1U> endRow;
186     int k;
187     coder::array<signed char, 1U> startCol;
```

```

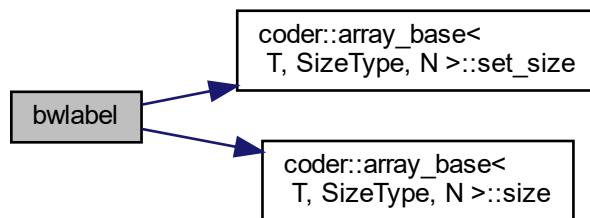
188 coder::array<int, 1U> labelForEachRun;
189 coder::array<int, 1U> labelsRenumbered;
190 numComponents = 0.0;
191 for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < 2542;
192     firstRunOnPreviousColumn++) {
193     im[firstRunOnPreviousColumn] = (varargin_1[firstRunOnPreviousColumn] != 0ULL);
194     L[firstRunOnPreviousColumn] = 0.0;
195 }
196 numRuns = 0;
197 for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 41;
198     lastRunOnPreviousColumn++) {
199     if (im[62 * lastRunOnPreviousColumn]) {
200         numRuns++;
201     }
202 }
203 for (k = 0; k < 61; k++) {
204     firstRunOnPreviousColumn = k + 62 * lastRunOnPreviousColumn;
205     if (im[firstRunOnPreviousColumn + 1] && (!im[firstRunOnPreviousColumn])) {
206         numRuns++;
207     }
208 }
209 }
210 }
211 if (numRuns != 0) {
212     int runCounter;
213     int row;
214     int firstRunOnThisColumn;
215     startRow.set_size(numRuns);
216     endRow.set_size(numRuns);
217     startCol.set_size(numRuns);
218     runCounter = 0;
219     for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 41;
220         lastRunOnPreviousColumn++) {
221         row = 1;
222         while (row <= 62) {
223             while ((row <= 62) && (!im[(row + 62 * lastRunOnPreviousColumn) - 1])) {
224                 row++;
225             }
226             if ((row <= 62) && im[(row + 62 * lastRunOnPreviousColumn) - 1]) {
227                 startCol[runCounter] = static_cast<signed char>
228                     (lastRunOnPreviousColumn + 1);
229                 startRow[runCounter] = static_cast<signed char>(row);
230                 while ((row <= 62) && im[(row + 62 * lastRunOnPreviousColumn) - 1]) {
231                     row++;
232                 }
233                 endRow[runCounter] = static_cast<signed char>(row - 1);
234                 runCounter++;
235             }
236         }
237     }
238 }
239 }
240 }
241 labelForEachRun.set_size(numRuns);
242 for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < numRuns;
243     firstRunOnPreviousColumn++) {
244     labelForEachRun[firstRunOnPreviousColumn] = 0;
245 }
246
247 k = 0;
248 runCounter = 1;
249 row = 1;
250 firstRunOnPreviousColumn = -1;
251 lastRunOnPreviousColumn = -1;
252 firstRunOnThisColumn = 0;
253 while (k + 1 <= numRuns) {
254     if (startCol[k] == runCounter + 1) {
255         firstRunOnPreviousColumn = firstRunOnThisColumn + 1;
256         firstRunOnThisColumn = k;
257         lastRunOnPreviousColumn = k;
258         runCounter = startCol[k];
259     } else {
260         if (startCol[k] > runCounter + 1) {
261             firstRunOnPreviousColumn = -1;
262             lastRunOnPreviousColumn = -1;
263             firstRunOnThisColumn = k;
264             runCounter = startCol[k];
265         }
266     }
267 }
268 if (firstRunOnPreviousColumn >= 0) {
269     for (int p = firstRunOnPreviousColumn - 1; p < lastRunOnPreviousColumn;
270         p++) {
271         if ((endRow[k] >= startRow[p] - 1) && (startRow[k] <= endRow[p] + 1))
272         {
273             if (labelForEachRun[k] == 0) {

```

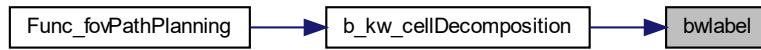
```

275         labelForEachRun[k] = labelForEachRun[p];
276         row++;
277     } else {
278         if (labelForEachRun[k] != labelForEachRun[p]) {
279             int root_k;
280             int root_p;
281             for (root_k = k; root_k + 1 != labelForEachRun[root_k]; root_k =
282                 labelForEachRun[root_k] - 1) {
283                 labelForEachRun[root_k] =
284                     labelForEachRun[labelForEachRun[root_k] - 1];
285             }
286
287             for (root_p = p; root_p + 1 != labelForEachRun[root_p]; root_p =
288                 labelForEachRun[root_p] - 1) {
289                 labelForEachRun[root_p] =
290                     labelForEachRun[labelForEachRun[root_p] - 1];
291             }
292
293             if (root_k + 1 != root_p + 1) {
294                 if (root_p + 1 < root_k + 1) {
295                     labelForEachRun[root_k] = root_p + 1;
296                     labelForEachRun[k] = root_p + 1;
297                 } else {
298                     labelForEachRun[root_p] = root_k + 1;
299                     labelForEachRun[p] = root_k + 1;
300                 }
301             }
302         }
303     }
304 }
305 }
306 }
307
308 if (labelForEachRun[k] == 0) {
309     labelForEachRun[k] = row;
310     row++;
311 }
312
313 k++;
314 }
315
316 labelsRenumbered.set_size(labelForEachRun.size());
317 for (k = 0; k < numRuns; k++) {
318     if (labelForEachRun[k] == k + 1) {
319         numComponents++;
320         labelsRenumbered[k] = static_cast<int>(numComponents);
321     }
322
323     labelsRenumbered[k] = labelsRenumbered[labelForEachRun[k] - 1];
324     firstRunOnPreviousColumn = startRow[k];
325     runCounter = endRow[k];
326     for (row = firstRunOnPreviousColumn; row <= runCounter; row++) {
327         L[(row + 62 * (startCol[k] - 1)) - 1] = labelsRenumbered[k];
328     }
329 }
330 }
331 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.1.3 c_bwlable()

```

void c_bwlable (
    const unsigned long long varargin_1_data[],
    const int varargin_1_size[2],
    double L_data[],
    int L_size[2] )
  
```

Definition at line 333 of file bwlable.cpp.

```

335 {
336     int firstRunOnPreviousColumn;
337     int lastRunOnPreviousColumn;
338     int runCounter;
339     int firstRunOnThisColumn;
340     double numComponents;
341     boolean_T im_data[3844];
342     int numRuns;
343     int p;
344     coder::array<signed char, 1U> startRow;
345     coder::array<signed char, 1U> endRow;
346     coder::array<signed char, 1U> startCol;
347     int k;
348     int row;
349     coder::array<int, 1U> labelForEachRun;
350     coder::array<int, 1U> labelsRenumbered;
351     firstRunOnPreviousColumn = varargin_1_size[0];
352     lastRunOnPreviousColumn = varargin_1_size[1];
353     runCounter = varargin_1_size[0] * varargin_1_size[1];
354     for (firstRunOnThisColumn = 0; firstRunOnThisColumn < runCounter;
355          firstRunOnThisColumn++) {
356         im_data[firstRunOnThisColumn] = (varargin_1_data[firstRunOnThisColumn] !=
357                                         OULL);
358     }
359     numComponents = 0.0;
360     L_size[0] = static_cast<signed char>(varargin_1_size[0]);
361     L_size[1] = static_cast<signed char>(varargin_1_size[1]);
362     runCounter = static_cast<signed char>(varargin_1_size[0]) * static_cast<signed
363                                         char>(varargin_1_size[1]);
364     if (0 <= runCounter - 1) {
365         std::memset(&L_data[0], 0, runCounter * sizeof(double));
366     }
367 }
368 numRuns = 0;
369 for (p = 0; p < lastRunOnPreviousColumn; p++) {
370     firstRunOnThisColumn = firstRunOnPreviousColumn * p;
371     if (im_data[firstRunOnThisColumn]) {
372         numRuns++;
373     }
374
375     for (k = 0; k <= firstRunOnPreviousColumn - 2; k++) {
376         row = k + firstRunOnThisColumn;
377         if (im_data[row + 1] && (!im_data[row])) {
378             numRuns++;
379         }
380     }
381 }
382 }
383 if (numRuns != 0) {
384     startRow.set_size(numRuns);
  
```

```

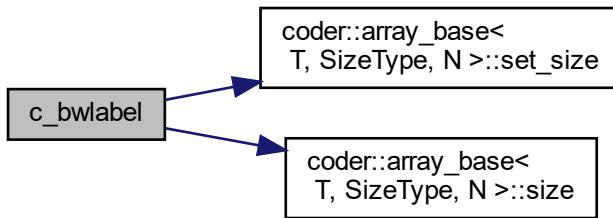
386     endRow.set_size(numRuns);
387     startCol.set_size(numRuns);
388     runCounter = 0;
389     for (p = 0; p < lastRunOnPreviousColumn; p++) {
390         row = 1;
391         while (row <= firstRunOnPreviousColumn) {
392             while ((row <= firstRunOnPreviousColumn) && (!im_data[(row +
393                 firstRunOnPreviousColumn * p) - 1])) {
394                 row++;
395             }
396             if ((row <= firstRunOnPreviousColumn) && im_data[(row +
397                 firstRunOnPreviousColumn * p) - 1]) {
398                 startCol[runCounter] = static_cast<signed char>(p + 1);
399                 startRow[runCounter] = static_cast<signed char>(row);
400                 while ((row <= firstRunOnPreviousColumn) && im_data[(row +
401                     firstRunOnPreviousColumn * p) - 1]) {
402                     row++;
403                 }
404             }
405             endRow[runCounter] = static_cast<signed char>(row - 1);
406             runCounter++;
407         }
408     }
409 }
410 }
411
412 labelForEachRun.set_size(numRuns);
413 for (firstRunOnThisColumn = 0; firstRunOnThisColumn < numRuns;
414     firstRunOnThisColumn++) {
415     labelForEachRun[firstRunOnThisColumn] = 0;
416 }
417
418 k = 0;
419 runCounter = 1;
420 row = 1;
421 firstRunOnPreviousColumn = -1;
422 lastRunOnPreviousColumn = -1;
423 firstRunOnThisColumn = 0;
424 while (k + 1 <= numRuns) {
425     if (startCol[k] == runCounter + 1) {
426         firstRunOnPreviousColumn = firstRunOnThisColumn + 1;
427         firstRunOnThisColumn = k;
428         lastRunOnPreviousColumn = k;
429         runCounter = startCol[k];
430     } else {
431         if (startCol[k] > runCounter + 1) {
432             firstRunOnPreviousColumn = -1;
433             lastRunOnPreviousColumn = -1;
434             firstRunOnThisColumn = k;
435             runCounter = startCol[k];
436         }
437     }
438
439     if (firstRunOnPreviousColumn >= 0) {
440         for (p = firstRunOnPreviousColumn - 1; p < lastRunOnPreviousColumn; p++)
441         {
442             if ((endRow[k] >= startRow[p] - 1) && (startRow[k] <= endRow[p] + 1))
443             {
444                 if (labelForEachRun[k] == 0) {
445                     labelForEachRun[k] = labelForEachRun[p];
446                     row++;
447                 } else {
448                     if (labelForEachRun[k] != labelForEachRun[p]) {
449                         int root_k;
450                         int root_p;
451                         for (root_k = k; root_k + 1 != labelForEachRun[root_k]; root_k =
452                             labelForEachRun[root_k] - 1) {
453                             labelForEachRun[root_k] =
454                             labelForEachRun[labelForEachRun[root_k] - 1];
455                         }
456
457                         for (root_p = p; root_p + 1 != labelForEachRun[root_p]; root_p =
458                             labelForEachRun[root_p] - 1) {
459                             labelForEachRun[root_p] =
460                             labelForEachRun[labelForEachRun[root_p] - 1];
461                         }
462
463                         if (root_k + 1 != root_p + 1) {
464                             if (root_p + 1 < root_k + 1) {
465                                 labelForEachRun[root_k] = root_p + 1;
466                                 labelForEachRun[k] = root_p + 1;
467                             } else {
468                                 labelForEachRun[root_p] = root_k + 1;
469                                 labelForEachRun[p] = root_k + 1;
470                             }
471                         }
472                     }
473                 }
474             }
475         }
476     }
477 }

```

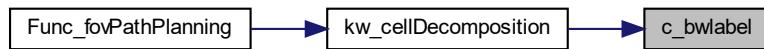
```

473         }
474     }
475   }
476 }
477
478   if (labelForEachRun[k] == 0) {
479     labelForEachRun[k] = row;
480     row++;
481   }
482
483   k++;
484 }
485
486 labelsRenumbered.set_size(labelForEachRun.size(0));
487 for (k = 0; k < numRuns; k++) {
488   if (labelForEachRun[k] == k + 1) {
489     numComponents++;
490     labelsRenumbered[k] = static_cast<int>(numComponents);
491   }
492
493   labelsRenumbered[k] = labelsRenumbered[labelForEachRun[k] - 1];
494   firstRunOnThisColumn = startRow[k];
495   row = endRow[k];
496   for (runCounter = firstRunOnThisColumn; runCounter <= row; runCounter++) {
497     L_data[(runCounter + L_size[0] * (startCol[k] - 1)) - 1] =
498       labelsRenumbered[k];
499   }
500 }
501 }
502 }
```

Here is the call graph for this function:



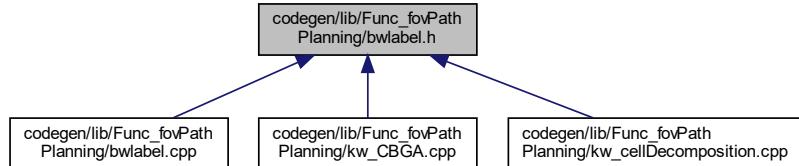
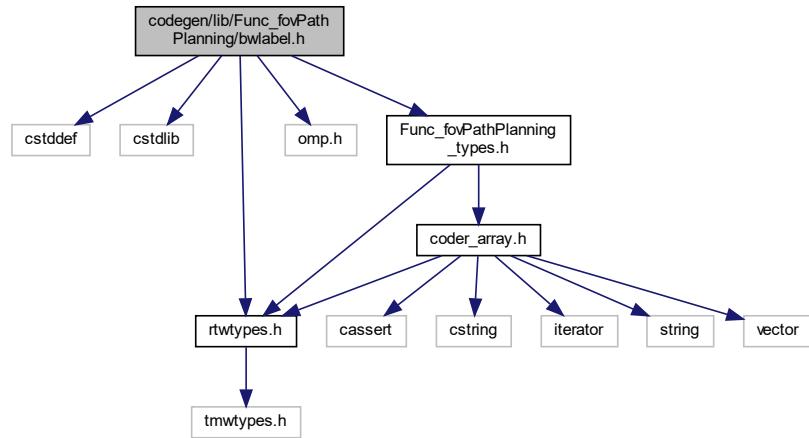
Here is the caller graph for this function:



7.6 codegen/lib/Func_fovPathPlanning/bwlabel.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
```

```
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for bwlabel.h:
```



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void b_bwlabel (const unsigned long long varargin_1[6000000], double L[6000000])`
- `void bwlabel (const unsigned long long varargin_1[2542], double L[2542])`
- `void c_bwlabel (const unsigned long long varargin_1_data[], const int varargin_1_size[2], double L_data[], int L_size[2])`

7.6.1 Macro Definition Documentation

7.6.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file bwlabel.h.

7.6.2 Function Documentation

7.6.2.1 b_bwlabel()

```
void b_bwlabel (
    const unsigned long long varargin_1[6000000],
    double L[6000000] )
```

Definition at line 20 of file bwlabel.cpp.

```
21 {
22     double numComponents;
23     int firstRunOnPreviousColumn;
24     int numRuns;
25     static boolean_T im[6000000];
26     int lastRunOnPreviousColumn;
27     coder::array<short, 1U> startRow;
28     coder::array<short, 1U> endRow;
29     int k;
30     coder::array<short, 1U> startCol;
31     coder::array<int, 1U> labelForEachRun;
32     coder::array<int, 1U> labelsRenumbered;
33     numComponents = 0.0;
34     for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < 6000000;
35          firstRunOnPreviousColumn++) {
36         im[firstRunOnPreviousColumn] = (varargin_1[firstRunOnPreviousColumn] != 0ULL);
37         L[firstRunOnPreviousColumn] = 0.0;
38     }
39
40     numRuns = 0;
41     for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 2000;
42          lastRunOnPreviousColumn++) {
43         if (im[3000 * lastRunOnPreviousColumn]) {
44             numRuns++;
45         }
46         for (k = 0; k < 2999; k++) {
47             firstRunOnPreviousColumn = k + 3000 * lastRunOnPreviousColumn;
48             if (im[firstRunOnPreviousColumn + 1] && (!im[firstRunOnPreviousColumn])) {
49                 numRuns++;
50             }
51         }
52     }
53 }
54
55 if (numRuns != 0) {
56     int runCounter;
57     int row;
58     int firstRunOnThisColumn;
59     startRow.set_size(numRuns);
60     endRow.set_size(numRuns);
61     startCol.set_size(numRuns);
62     runCounter = 0;
63     for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 2000;
64          lastRunOnPreviousColumn++) {
65         row = 1;
66         while (row <= 3000) {
67             while ((row <= 3000) && (!im[(row + 3000 * lastRunOnPreviousColumn) - 1]))
68             {
69                 row++;
70             }
71             if ((row <= 3000) && im[(row + 3000 * lastRunOnPreviousColumn) - 1]) {
72                 startCol[runCounter] = static_cast<short>(lastRunOnPreviousColumn + 1);
73                 startRow[runCounter] = static_cast<short>(row);
74                 while ((row <= 3000) && im[(row + 3000 * lastRunOnPreviousColumn) - 1])
75                 {
76
```

```

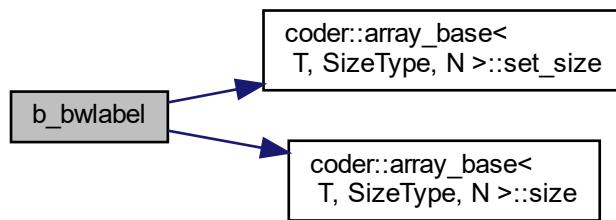
77         row++;
78     }
79
80     endRow[runCounter] = static_cast<short>(row - 1);
81     runCounter++;
82   }
83 }
84 }
85
86 labelForEachRun.set_size(numRuns);
87 for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < numRuns;
88       firstRunOnPreviousColumn++) {
89   labelForEachRun[firstRunOnPreviousColumn] = 0;
90 }
91
92 k = 0;
93 runCounter = 1;
94 row = 1;
95 firstRunOnPreviousColumn = -1;
96 lastRunOnPreviousColumn = -1;
97 firstRunOnThisColumn = 0;
98 while (k + 1 <= numRuns) {
99   if (startCol[k] == runCounter + 1) {
100     firstRunOnPreviousColumn = firstRunOnThisColumn + 1;
101     firstRunOnThisColumn = k;
102     lastRunOnPreviousColumn = k;
103     runCounter = startCol[k];
104   } else {
105     if (startCol[k] > runCounter + 1) {
106       firstRunOnPreviousColumn = -1;
107       lastRunOnPreviousColumn = -1;
108       firstRunOnThisColumn = k;
109       runCounter = startCol[k];
110     }
111   }
112
113   if (firstRunOnPreviousColumn >= 0) {
114     for (int p = firstRunOnPreviousColumn - 1; p < lastRunOnPreviousColumn;
115          p++) {
116       if ((endRow[k] >= startRow[p] - 1) && (startRow[k] <= endRow[p] + 1))
117     {
118       if (labelForEachRun[k] == 0) {
119         labelForEachRun[k] = labelForEachRun[p];
120         row++;
121       } else {
122         if (labelForEachRun[k] != labelForEachRun[p]) {
123           int root_k;
124           int root_p;
125           for (root_k = k; root_k + 1 != labelForEachRun[root_k]; root_k =
126                 labelForEachRun[root_k] - 1) {
127             labelForEachRun[root_k] =
128               labelForEachRun[labelForEachRun[root_k] - 1];
129           }
130
131           for (root_p = p; root_p + 1 != labelForEachRun[root_p]; root_p =
132                 labelForEachRun[root_p] - 1) {
133             labelForEachRun[root_p] =
134               labelForEachRun[labelForEachRun[root_p] - 1];
135           }
136
137           if (root_k + 1 != root_p + 1) {
138             if (root_p + 1 < root_k + 1) {
139               labelForEachRun[root_k] = root_p + 1;
140               labelForEachRun[k] = root_p + 1;
141             } else {
142               labelForEachRun[root_p] = root_k + 1;
143               labelForEachRun[p] = root_k + 1;
144             }
145           }
146         }
147       }
148     }
149   }
150 }
151
152   if (labelForEachRun[k] == 0) {
153     labelForEachRun[k] = row;
154     row++;
155   }
156
157   k++;
158 }
159
160 labelsRenumbered.set_size(labelForEachRun.size(0));
161 for (k = 0; k < numRuns; k++) {
162   if (labelForEachRun[k] == k + 1) {
163     numComponents++;

```

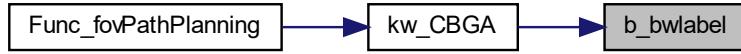
```

164     labelsRenumbered[k] = static_cast<int>(numComponents);
165 }
166
167 labelsRenumbered[k] = labelsRenumbered[labelForEachRun[k] - 1];
168 firstRunOnPreviousColumn = startRow[k];
169 runCounter = endRow[k];
170 for (row = firstRunOnPreviousColumn; row <= runCounter; row++) {
171     L[(row + 3000 * (startCol[k] - 1)) - 1] = labelsRenumbered[k];
172 }
173 }
174 }
175 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.2.2 **bwlabel()**

```
void bwlabel (
    const unsigned long long varargin_1[2542],
    double L[2542] )
```

Definition at line 177 of file bwlabel.cpp.

```

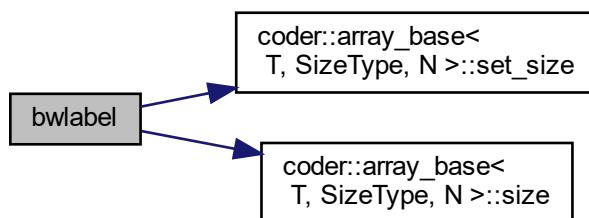
178 {
179     double numComponents;
180     int firstRunOnPreviousColumn;
181     int numRuns;
182     boolean_T im[2542];
183     int lastRunOnPreviousColumn;
184     coder::array<signed char, 1U> startRow;
185     coder::array<signed char, 1U> endRow;
186     int k;
187     coder::array<signed char, 1U> startCol;
188     coder::array<int, 1U> labelForEachRun;
```

```
189  coder::array<int, 1U> labelsRenumbered;
190  numComponents = 0.0;
191  for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < 2542;
192      firstRunOnPreviousColumn++) {
193      im[firstRunOnPreviousColumn] = (varargin_1[firstRunOnPreviousColumn] != 0ULL);
194      L[firstRunOnPreviousColumn] = 0.0;
195  }
196
197  numRuns = 0;
198  for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 41;
199      lastRunOnPreviousColumn++) {
200      if (im[62 * lastRunOnPreviousColumn]) {
201          numRuns++;
202      }
203
204      for (k = 0; k < 61; k++) {
205          firstRunOnPreviousColumn = k + 62 * lastRunOnPreviousColumn;
206          if (im[firstRunOnPreviousColumn + 1] && (!im[firstRunOnPreviousColumn])) {
207              numRuns++;
208          }
209      }
210  }
211
212  if (numRuns != 0) {
213      int runCounter;
214      int row;
215      int firstRunOnThisColumn;
216      startRow.set_size(numRuns);
217      endRow.set_size(numRuns);
218      startCol.set_size(numRuns);
219      runCounter = 0;
220      for (lastRunOnPreviousColumn = 0; lastRunOnPreviousColumn < 41;
221          lastRunOnPreviousColumn++) {
222          row = 1;
223          while (row <= 62) {
224              while ((row <= 62) && (!im[(row + 62 * lastRunOnPreviousColumn) - 1])) {
225                  row++;
226              }
227
228              if ((row <= 62) && im[(row + 62 * lastRunOnPreviousColumn) - 1]) {
229                  startCol[runCounter] = static_cast<signed char>
230                      (lastRunOnPreviousColumn + 1);
231                  startRow[runCounter] = static_cast<signed char>(row);
232                  while ((row <= 62) && im[(row + 62 * lastRunOnPreviousColumn) - 1]) {
233                      row++;
234                  }
235
236                  endRow[runCounter] = static_cast<signed char>(row - 1);
237                  runCounter++;
238              }
239          }
240      }
241
242      labelForEachRun.set_size(numRuns);
243      for (firstRunOnPreviousColumn = 0; firstRunOnPreviousColumn < numRuns;
244          firstRunOnPreviousColumn++) {
245          labelForEachRun[firstRunOnPreviousColumn] = 0;
246      }
247
248      k = 0;
249      runCounter = 1;
250      row = 1;
251      firstRunOnPreviousColumn = -1;
252      lastRunOnPreviousColumn = -1;
253      firstRunOnThisColumn = 0;
254      while (k + 1 <= numRuns) {
255          if (startCol[k] == runCounter + 1) {
256              firstRunOnPreviousColumn = firstRunOnThisColumn + 1;
257              firstRunOnThisColumn = k;
258              lastRunOnPreviousColumn = k;
259              runCounter = startCol[k];
260          } else {
261              if (startCol[k] > runCounter + 1) {
262                  firstRunOnPreviousColumn = -1;
263                  lastRunOnPreviousColumn = -1;
264                  firstRunOnThisColumn = k;
265                  runCounter = startCol[k];
266              }
267          }
268
269          if (firstRunOnPreviousColumn >= 0) {
270              for (int p = firstRunOnPreviousColumn - 1; p < lastRunOnPreviousColumn;
271                  p++) {
272                  if ((endRow[k] >= startRow[p] - 1) && (startRow[k] <= endRow[p] + 1))
273                  {
274                      if (labelForEachRun[k] == 0) {
275                          labelForEachRun[k] = labelForEachRun[p];
```

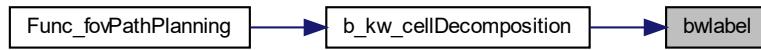
```

276         row++;
277     } else {
278         if (labelForEachRun[k] != labelForEachRun[p]) {
279             int root_k;
280             int root_p;
281             for (root_k = k; root_k + 1 != labelForEachRun[root_k]; root_k =
282                 labelForEachRun[root_k] - 1) {
283                 labelForEachRun[root_k] =
284                     labelForEachRun[labelForEachRun[root_k] - 1];
285             }
286
287             for (root_p = p; root_p + 1 != labelForEachRun[root_p]; root_p =
288                 labelForEachRun[root_p] - 1) {
289                 labelForEachRun[root_p] =
290                     labelForEachRun[labelForEachRun[root_p] - 1];
291             }
292
293             if (root_k + 1 != root_p + 1) {
294                 if (root_p + 1 < root_k + 1) {
295                     labelForEachRun[root_k] = root_p + 1;
296                     labelForEachRun[k] = root_p + 1;
297                 } else {
298                     labelForEachRun[root_p] = root_k + 1;
299                     labelForEachRun[p] = root_k + 1;
300                 }
301             }
302         }
303     }
304 }
305 }
306 }
307
308 if (labelForEachRun[k] == 0) {
309     labelForEachRun[k] = row;
310     row++;
311 }
312
313 k++;
314 }
315
316 labelsRenumbered.set_size(labelForEachRun.size());
317 for (k = 0; k < numRuns; k++) {
318     if (labelForEachRun[k] == k + 1) {
319         numComponents++;
320         labelsRenumbered[k] = static_cast<int>(numComponents);
321     }
322
323     labelsRenumbered[k] = labelsRenumbered[labelForEachRun[k] - 1];
324     firstRunOnPreviousColumn = startRow[k];
325     runCounter = endRow[k];
326     for (row = firstRunOnPreviousColumn; row <= runCounter; row++) {
327         L[(row + 62 * (startCol[k] - 1)) - 1] = labelsRenumbered[k];
328     }
329 }
330 }
331 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.2.3 c_bwlabel()

```

void c_bwlabel (
    const unsigned long long varargin_1_data[],
    const int varargin_1_size[2],
    double L_data[],
    int L_size[2] )
  
```

Definition at line 333 of file bwlabel.cpp.

```

335 {
336     int firstRunOnPreviousColumn;
337     int lastRunOnPreviousColumn;
338     int runCounter;
339     int firstRunOnThisColumn;
340     double numComponents;
341     boolean_T im_data[3844];
342     int numRuns;
343     int p;
344     coder::array<signed char, 1U> startRow;
345     coder::array<signed char, 1U> endRow;
346     coder::array<signed char, 1U> startCol;
347     int k;
348     int row;
349     coder::array<int, 1U> labelForEachRun;
350     coder::array<int, 1U> labelsRenumbered;
351     firstRunOnPreviousColumn = varargin_1_size[0];
352     lastRunOnPreviousColumn = varargin_1_size[1];
353     runCounter = varargin_1_size[0] * varargin_1_size[1];
354     for (firstRunOnThisColumn = 0; firstRunOnThisColumn < runCounter;
355          firstRunOnThisColumn++) {
356         im_data[firstRunOnThisColumn] = (varargin_1_data[firstRunOnThisColumn] !=
357                                         OULL);
358     }
359     numComponents = 0.0;
360     L_size[0] = static_cast<signed char>(varargin_1_size[0]);
361     L_size[1] = static_cast<signed char>(varargin_1_size[1]);
362     runCounter = static_cast<signed char>(varargin_1_size[0]) * static_cast<signed
363                                         char>(varargin_1_size[1]);
364     if (0 <= runCounter - 1) {
365         std::memset(&L_data[0], 0, runCounter * sizeof(double));
366     }
367 }
368 numRuns = 0;
369 for (p = 0; p < lastRunOnPreviousColumn; p++) {
370     firstRunOnThisColumn = firstRunOnPreviousColumn * p;
371     if (im_data[firstRunOnThisColumn]) {
372         numRuns++;
373     }
374
375     for (k = 0; k <= firstRunOnPreviousColumn - 2; k++) {
376         row = k + firstRunOnThisColumn;
377         if (im_data[row + 1] && (!im_data[row])) {
378             numRuns++;
379         }
380     }
381 }
382 }
383 if (numRuns != 0) {
384     startRow.set_size(numRuns);
  
```

```

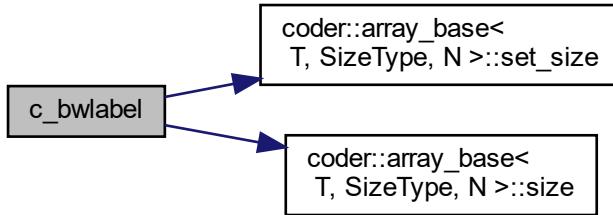
386     endRow.set_size(numRuns);
387     startCol.set_size(numRuns);
388     runCounter = 0;
389     for (p = 0; p < lastRunOnPreviousColumn; p++) {
390         row = 1;
391         while (row <= firstRunOnPreviousColumn) {
392             while ((row <= firstRunOnPreviousColumn) && (!im_data[(row +
393                 firstRunOnPreviousColumn * p) - 1])) {
394                 row++;
395             }
396             if ((row <= firstRunOnPreviousColumn) && im_data[(row +
397                 firstRunOnPreviousColumn * p) - 1]) {
398                 startCol[runCounter] = static_cast<signed char>(p + 1);
399                 startRow[runCounter] = static_cast<signed char>(row);
400                 while ((row <= firstRunOnPreviousColumn) && im_data[(row +
401                     firstRunOnPreviousColumn * p) - 1]) {
402                     row++;
403                 }
404             }
405             endRow[runCounter] = static_cast<signed char>(row - 1);
406             runCounter++;
407         }
408     }
409 }
410 }
411
412 labelForEachRun.set_size(numRuns);
413 for (firstRunOnThisColumn = 0; firstRunOnThisColumn < numRuns;
414     firstRunOnThisColumn++) {
415     labelForEachRun[firstRunOnThisColumn] = 0;
416 }
417
418 k = 0;
419 runCounter = 1;
420 row = 1;
421 firstRunOnPreviousColumn = -1;
422 lastRunOnPreviousColumn = -1;
423 firstRunOnThisColumn = 0;
424 while (k + 1 <= numRuns) {
425     if (startCol[k] == runCounter + 1) {
426         firstRunOnPreviousColumn = firstRunOnThisColumn + 1;
427         firstRunOnThisColumn = k;
428         lastRunOnPreviousColumn = k;
429         runCounter = startCol[k];
430     } else {
431         if (startCol[k] > runCounter + 1) {
432             firstRunOnPreviousColumn = -1;
433             lastRunOnPreviousColumn = -1;
434             firstRunOnThisColumn = k;
435             runCounter = startCol[k];
436         }
437     }
438
439     if (firstRunOnPreviousColumn >= 0) {
440         for (p = firstRunOnPreviousColumn - 1; p < lastRunOnPreviousColumn; p++)
441         {
442             if ((endRow[k] >= startRow[p] - 1) && (startRow[k] <= endRow[p] + 1))
443             {
444                 if (labelForEachRun[k] == 0) {
445                     labelForEachRun[k] = labelForEachRun[p];
446                     row++;
447                 } else {
448                     if (labelForEachRun[k] != labelForEachRun[p]) {
449                         int root_k;
450                         int root_p;
451                         for (root_k = k; root_k + 1 != labelForEachRun[root_k]; root_k =
452                             labelForEachRun[root_k] - 1) {
453                             labelForEachRun[root_k] =
454                             labelForEachRun[labelForEachRun[root_k] - 1];
455                         }
456
457                         for (root_p = p; root_p + 1 != labelForEachRun[root_p]; root_p =
458                             labelForEachRun[root_p] - 1) {
459                             labelForEachRun[root_p] =
460                             labelForEachRun[labelForEachRun[root_p] - 1];
461                         }
462
463                         if (root_k + 1 != root_p + 1) {
464                             if (root_p + 1 < root_k + 1) {
465                                 labelForEachRun[root_k] = root_p + 1;
466                                 labelForEachRun[k] = root_p + 1;
467                             } else {
468                                 labelForEachRun[root_p] = root_k + 1;
469                                 labelForEachRun[p] = root_k + 1;
470                             }
471                         }
472                     }
473                 }
474             }
475         }
476     }
477 }

```

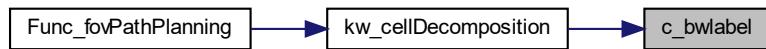
```

473         }
474     }
475   }
476 }
477
478   if (labelForEachRun[k] == 0) {
479     labelForEachRun[k] = row;
480     row++;
481   }
482
483   k++;
484 }
485
486 labelsRenumbered.set_size(labelForEachRun.size(0));
487 for (k = 0; k < numRuns; k++) {
488   if (labelForEachRun[k] == k + 1) {
489     numComponents++;
490     labelsRenumbered[k] = static_cast<int>(numComponents);
491   }
492
493   labelsRenumbered[k] = labelsRenumbered[labelForEachRun[k] - 1];
494   firstRunOnThisColumn = startRow[k];
495   row = endRow[k];
496   for (runCounter = firstRunOnThisColumn; runCounter <= row; runCounter++) {
497     L_data[(runCounter + L_size[0] * (startCol[k] - 1)) - 1] =
498       labelsRenumbered[k];
499   }
500 }
501 }
502 }
```

Here is the call graph for this function:



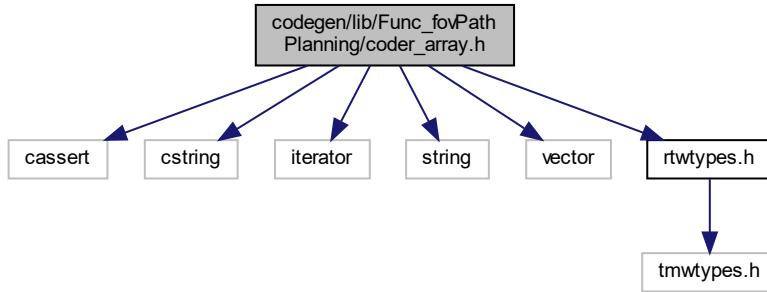
Here is the caller graph for this function:



7.7 codegen/lib/Func_fovPathPlanning(coder_array.h File Reference)

```
#include <cassert>
#include <cstring>
#include <iterator>
```

```
#include <string>
#include <vector>
#include "rtwtypes.h"
Include dependency graph for coder_array.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `coder::detail::data_ptr< T, SZ >`
- class `coder::array_iterator< T >`
- class `coder::const_array_iterator< T >`
- class `coder::detail::numel< N >`
- class `coder::detail::numel< 0 >`
- class `coder::detail::index_nd< I >`
- class `coder::detail::index_nd< 0 >`
- struct `coder::detail::match_dimensions< Cond >`
- struct `coder::detail::match_dimensions< true >`
- class `coder::array_base< T, SZ, N >`
- class `coder::array< T, N >`
- class `coder::array< char_T, 2 >`
- class `coder::array< T, 2 >`
- class `coder::array< T, 1 >`

Namespaces

- `coder`
- `coder::detail`

Macros

- #define _mw_coder_array_h
- #define CODER_ARRAY_NEW_DELETE
- #define CODER_NEW(T, N) new T[N]
- #define CODER_DELETE(P) delete[](P)

Typedefs

- typedef int32_T coder::SizeType

7.7.1 Macro Definition Documentation

7.7.1.1 _mw_coder_array_h

```
#define _mw_coder_array_h
```

Definition at line 5 of file coder_array.h.

7.7.1.2 CODER_ARRAY_NEW_DELETE

```
#define CODER_ARRAY_NEW_DELETE
```

Definition at line 55 of file coder_array.h.

7.7.1.3 CODER_DELETE

```
#define CODER_DELETE(
    P ) delete[ ](P)
```

Definition at line 57 of file coder_array.h.

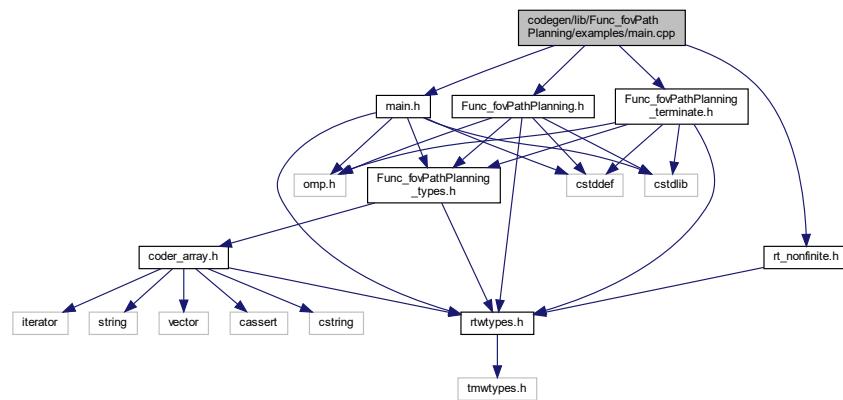
7.7.1.4 CODER_NEW

```
#define CODER_NEW(
    T,
    N ) new T[N]
```

Definition at line 56 of file coder_array.h.

7.8 codegen/lib(Func_fovPathPlanning/examples/main.cpp File Reference)

```
#include "main.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_terminate.h"
#include "rt_nonfinite.h"
Include dependency graph for main.cpp:
```



Functions

- int `main` (int, const char *const [])

7.8.1 Function Documentation

7.8.1.1 main()

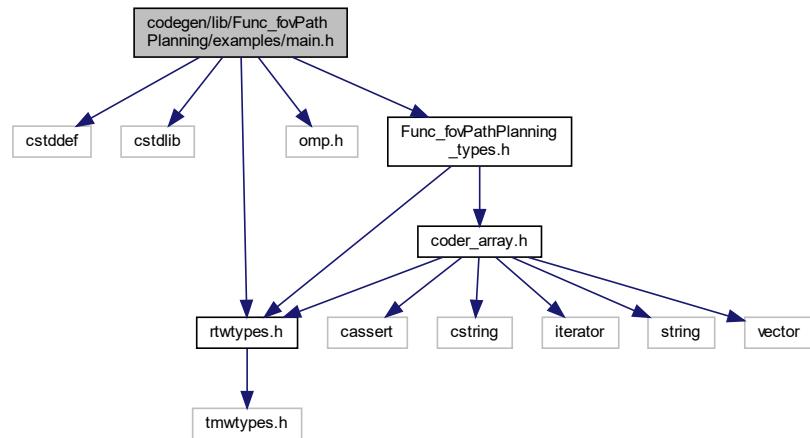
```
int main (
    int ,
    const char * const [ ] )
```

Definition at line 133 of file main.cpp.

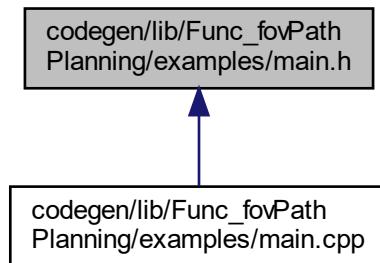
```
134 {
135     // The initialize function is being called automatically from your entry-point function. So, a call to
136     // initialize is not included here.
137     // Invoke the entry-point functions.
138     // You can call entry-point functions multiple times.
139     main_Func_fovPathPlanning();
140
141     // Terminate the application.
142     // You do not need to do this more than one time.
143     Func_fovPathPlanning_terminate();
144 }
```

7.9 codegen/lib/Func_fovPathPlanning/examples/main.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for main.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define MAX_THREADS omp_get_max_threads()

Functions

- int main (int argc, const char *const argv[])

7.9.1 Macro Definition Documentation

7.9.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 45 of file main.h.

7.9.2 Function Documentation

7.9.2.1 main()

```
int main (
    int argc,
    const char *const argv[] )
```

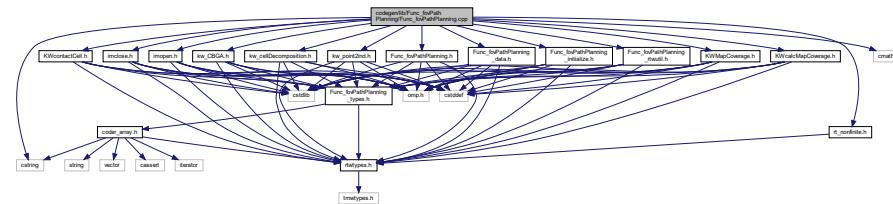
Definition at line 133 of file main.cpp.

```
134 {
135     // The initialize function is being called automatically from your entry-point function. So, a call to
136     // initialize is not included here.
137     // Invoke the entry-point functions.
138     // You can call entry-point functions multiple times.
139     main_Func_fovPathPlanning();
140     // Terminate the application.
141     // You do not need to do this more than one time.
142     Func_fovPathPlanning_terminate();
143     return 0;
144 }
```

7.10 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning.cpp File Reference

```
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_data.h"
#include "Func_fovPathPlanning_initialize.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "KWMapCoverage.h"
#include "KWCalcMapCoverage.h"
#include "KWcontactCell.h"
#include "imclose.h"
#include "imopen.h"
#include "kw_CBGA.h"
#include "kw_cellDecomposition.h"
#include "kw_point2ind.h"
#include "rt_nonfinite.h"
#include <cmath>
```

```
#include <cstring>
Include dependency graph for Func_fovPathPlanning.cpp:
```



Functions

- void [Func_fovPathPlanning](#) (long long, const unsigned long long binaryMap[6000000], const [cell_wrap_0](#) coordix[3], const [cell_wrap_0](#) coordiy[3])

7.10.1 Function Documentation

7.10.1.1 Func_fovPathPlanning()

```
void Func_fovPathPlanning (
    long long ,
    const unsigned long long binaryMap[6000000],
    const cell\_wrap\_0 coordix[3],
    const cell\_wrap\_0 coordiy[3] )
```

Definition at line 30 of file Func_fovPathPlanning.cpp.

```
32 {
33     int k;
34     int nx;
35     cell\_wrap\_1 ccoordiny[3];
36     cell\_wrap\_1 ccoordinx[3];
37     static unsigned long long coveredMap[6000000];
38     static unsigned long long b_binaryMap[6000000];
39     coder::array<double, 2U> x;
40     struct_T sensorParam;
41     cell\_wrap\_1 cluster[3];
42     cell\_wrap\_1 cell_index[3];
43     double num;
44     double prev_cell[6];
45     double nearClusterNum;
46     double b_robot;
47     double node;
48     double point[2];
49     double phase;
50     boolean_T b_x[6];
51     double unusedExpr[200];
52     if (!isInitialized_Func_fovPathPlanning) {
53         Func_fovPathPlanning_initialize();
54     }
55
56 //     img_name = 'map_outline.jpg';
57 //     load('kumohResults.mat');
58 // [coordix, coordiy, coorditheta, robotPoses] = Func_fovPathPlanning(img_name, coordix, coordiy,
59 // show_flag);
60 // icoordinx = coordix;
61 // icoordiny = coordiy;
62 for (k = 0; k < 3; k++) {
63     ccoordiny[k].f1.set_size(1, 10);
64     ccoordinx[k].f1.set_size(1, 10);
65     for (nx = 0; nx < 10; nx++) {
```

```

66     ccoordiy[k].f1[nx] = coordiy[k].f1[nx];
67     ccoordix[k].f1[nx] = coordix[k].f1[nx];
68 }
69 }
70
71 for (nx = 0; nx < 6000000; nx++) {
72     unsigned long long u;
73     u = binaryMap[nx];
74     b_binaryMap[nx] = binaryMap[nx];
75     if (binaryMap[nx] < 230ULL) {
76         u = OULL;
77         b_binaryMap[nx] = OULL;
78     }
79
80     if (u >= 230ULL) {
81         b_binaryMap[nx] = 255ULL;
82     }
83 }
84
85 //      se = offsetstrel('ball',5,3);
86 std::memcpy(&coveredMap[0], &b_binaryMap[0], 6000000U * sizeof(unsigned long
87     long));
88 imclose(coveredMap, b_binaryMap);
89 std::memcpy(&coveredMap[0], &b_binaryMap[0], 6000000U * sizeof(unsigned long
90     long));
91 imopen(coveredMap, b_binaryMap);
92
93 //      coveredMap = cat(3, binaryMap, binaryMap, binaryMap);
94 for (int robot = 0; robot < 5; robot++) {
95     if (robot + 1 > 3) {
96         ccoordix[robot].f1.set_size(1, 1);
97         ccoordix[robot].f1[0] = rtNaN;
98         ccoordiy[robot].f1.set_size(1, 1);
99         ccoordiy[robot].f1[0] = rtNaN;
100    } else {
101        x.set_size(ccoordix[robot].f1.size(0), ccoordix[robot].f1.size(1));
102        nx = ccoordix[robot].f1.size(0) * ccoordix[robot].f1.size(1);
103        for (k = 0; k < nx; k++) {
104            x[k] = ccoordix[robot].f1[k];
105        }
106
107        nx = ccoordix[robot].f1.size(1);
108        for (k = 0; k < nx; k++) {
109            x[k] = rt_rounnd_snf(x[k]);
110        }
111
112        ccoordix[robot].f1.set_size(1, x.size(1));
113        nx = x.size(0) * x.size(1);
114        for (k = 0; k < nx; k++) {
115            ccoordix[robot].f1[k] = x[k];
116        }
117
118        x.set_size(ccordiy[robot].f1.size(0), ccoordiy[robot].f1.size(1));
119        nx = ccoordiy[robot].f1.size(0) * ccoordiy[robot].f1.size(1);
120        for (k = 0; k < nx; k++) {
121            x[k] = ccoordiy[robot].f1[k];
122        }
123
124        nx = ccoordiy[robot].f1.size(1);
125        for (k = 0; k < nx; k++) {
126            x[k] = rt_rounnd_snf(x[k]);
127        }
128
129        ccoordiy[robot].f1.set_size(1, x.size(1));
130        nx = x.size(0) * x.size(1);
131        for (k = 0; k < nx; k++) {
132            ccoordiy[robot].f1[k] = x[k];
133        }
134    }
135 }
136
137 // imshow(binaryMap);
138 sensorParam.fov = 1.2217304763960306;
139 sensorParam.dFOV = 0.0043633231299858239;
140 sensorParam.radius = 300.0;
141 sensorParam.dvec = 15.0;
142 sensorParam.scolor = 100.0;
143 sensorParam.dc = 1.0;
144 sensorParam.Rmin = 75.0;
145 sensorParam.size[0] = 3000.0;
146 sensorParam.size[1] = 2000.0;
147 sensorParam.cs = 49.0;
148 sensorParam.mindis = 150.0;
149 sensorParam.H = 3000.0;
150 sensorParam.W = 2000.0;
151
152 KWcalcMapCoverage(b_binaryMap, ccoordix, ccoordiy, coveredMap);

```

```

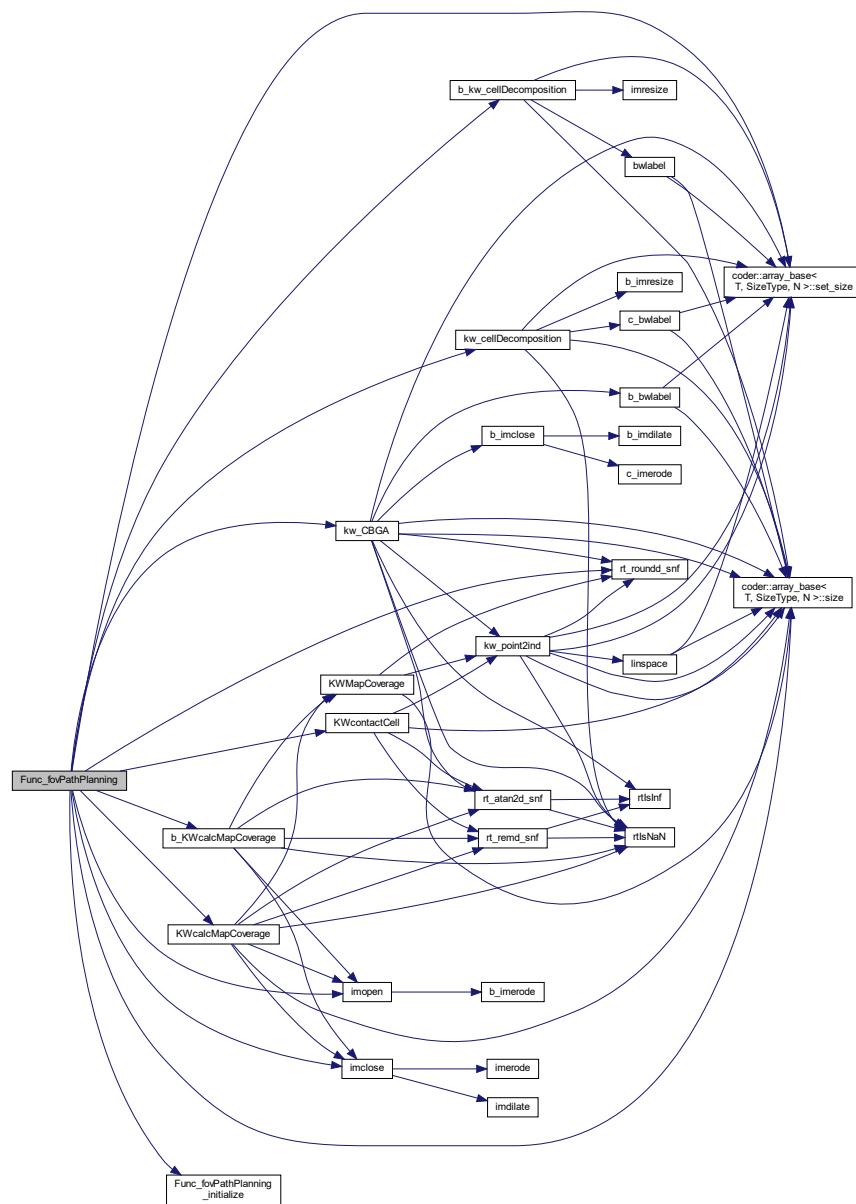
155 //      kw_imshow(ccoordix, ccoordiy);
156 //      kw_dispCoveragePercent();
157 // % % % % % % % % % % % % % % % % % % % % % %
158 cluster[0] = ccoordix[0];
159 cell_index[0] = ccoordiy[0];
160 cluster[1] = ccoordix[1];
161 cell_index[1] = ccoordiy[1];
162 cluster[2] = ccoordix[2];
163 cell_index[2] = ccoordiy[2];
164 num = b_kw_cellDecomposition(coveredMap, cluster, cell_index);
165 for (k = 0; k < 6; k++) {
166     prev_cell[k] = 0.0;
167 }
168
169
170 int exitg1;
171 do {
172     exitg1 = 0;
173     KWcontactCell(b_binaryMap, cluster, num, ccoordix, ccoordiy, &nearClusterNum,
174                     &b_robot, &node, point, &phase);
175     if (nearClusterNum == 0.0) {
176         exitg1 = 1;
177     } else {
178         boolean_T y;
179         boolean_T exitg2;
180         b_x[0] = (nearClusterNum != prev_cell[0]);
181         b_x[1] = (b_robot != prev_cell[1]);
182         b_x[2] = (node != prev_cell[2]);
183         b_x[3] = (point[0] != prev_cell[3]);
184         b_x[4] = (point[1] != prev_cell[4]);
185         b_x[5] = (phase != prev_cell[5]);
186         y = false;
187         k = 0;
188         exitg2 = false;
189         while ((!exitg2) && (k < 6)) {
190             if (!b_x[k]) {
191                 k++;
192             } else {
193                 y = true;
194                 exitg2 = true;
195             }
196         }
197     }
198
199     if (!y) {
200         double point_idx_1;
201         num = point[0] + 150.0 * std::cos(phase);
202         point_idx_1 = point[1] + 150.0 * std::sin(phase);
203         point[0] = num;
204         point[1] = point_idx_1;
205     }
206
207     prev_cell[0] = nearClusterNum;
208     prev_cell[1] = b_robot;
209     prev_cell[2] = node;
210     prev_cell[3] = point[0];
211     prev_cell[4] = point[1];
212     prev_cell[5] = phase;
213     kw_CBGA(&sensorParam, coveredMap, b_binaryMap, point, phase, unusedExpr);
214
215     //      coordix{robot} = [ccoordix{robot}(1:node-1), path(1,1:8),
216     ccoordix{robot}(node+1:end)];
217     //      coordiy{robot} = [ccoordiy{robot}(1:node-1), path(2,1:8),
218     ccoordiy{robot}(node+1:end)];
219     kw_cellDecomposition(coveredMap, coordix, coordiy, cluster, cell_index,
220                         &num);
221
222     //      function [cells, cell_index, numCluster] =
223     kw_cellDecomposition(sensorParam,coveredMap,coordix, coordiy, showing_flag)
224 }
225 //      new covered calculate
226 //      for robot = 1:5
227 //          if isnan(coordix{robot})
228 //              continue;
229 //          end
230 //          fx = coordix{robot};
231 //          fy = coordiy{robot};
232 //          ix = icoordix{robot};
233 //          iy = icoordiy{robot};
234 //          coordix{robot} = [fx(:,1:end), fliplr(ix(:,1:end-1))];
235 //          coordiy{robot} = [fy(:,1:end), fliplr(iy(:,1:end-1))];
236 b_KWcalcMapCoverage(b_binaryMap, &sensorParam, coordix, coordiy, coveredMap);
237
238 //      coverager Mapbinary Map3DoF  
239 //      function [coveredMap]= KWcalcMapCoverage(binaryMap,sensorParam,coordix, coordiy,
```

```

    show_flag)
240 //      [robotPoses, coorditheta] = kw_imshow(coordix, coordiy);
241 //      kw_dispCoveragePercent();
242 //      [coordix, coordiy, coorditheta, robotPoses] = KWdataPreProcessing2(coordix, coordiy,
243 //      coorditheta, robotPoses);
243 //      save('PrevkwResults.mat', 'coordix', 'coordiy', 'coorditheta', 'robotPoses');
244 }

```

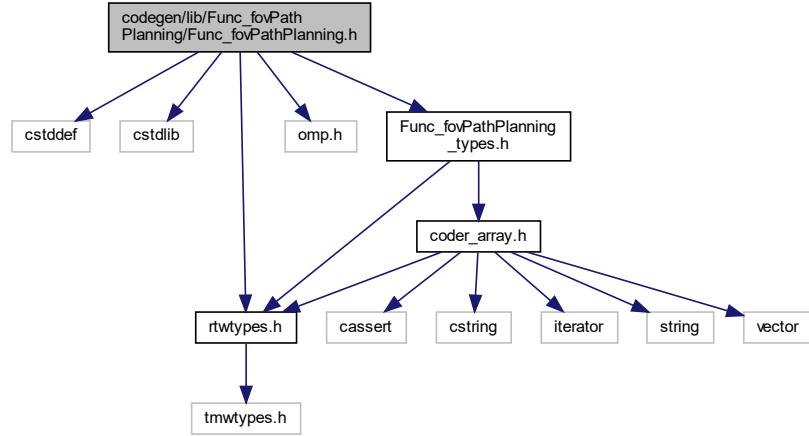
Here is the call graph for this function:



7.11 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning.h File Reference

```
#include <cstdint>
#include <cstdlib>
```

```
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for Func_fovPathPlanning.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void Func_fovPathPlanning (long long size_img, const unsigned long long binaryMap[6000000], const cell_wrap_0 coordix[3], const cell_wrap_0 coordiy[3])`

7.11.1 Macro Definition Documentation

7.11.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file `Func_fovPathPlanning.h`.

7.11.2 Function Documentation

7.11.2.1 Func_fovPathPlanning()

```
void Func_fovPathPlanning (
    long long size_img,
    const unsigned long long binaryMap[6000000],
    const cell_wrap_0 coordix[3],
    const cell_wrap_0 coordiy[3] )
```

Definition at line 30 of file Func_fovPathPlanning.cpp.

```
32 {
33     int k;
34     int nx;
35     cell_wrap_1 ccoordiy[3];
36     cell_wrap_1 ccoordix[3];
37     static unsigned long long coveredMap[6000000];
38     static unsigned long long b_binaryMap[6000000];
39     coder::array<double, 2U> x;
40     struct_T sensorParam;
41     cell_wrap_1 cluster[3];
42     cell_wrap_1 cell_index[3];
43     double num;
44     double prev_cell[6];
45     double nearClusterNum;
46     double b_robot;
47     double node;
48     double point[2];
49     double phase;
50     boolean_T b_x[6];
51     double unusedExpr[200];
52     if (!isInitialized_Func_fovPathPlanning) {
53         Func_fovPathPlanning_initialize();
54     }
55
56     //     img_name = 'map_outline.jpg';
57     //     load('kumohResults.mat');
58     // [coordix, coordiy, coorditheta, robotPoses] = Func_fovPathPlanning(img_name, coordix, coordiy,
59     // show_flag);
60     // icoordix = coordix;
61     // icoordiy = coordiy;
62     // %% t  y 
63     for (k = 0; k < 3; k++) {
64         ccoordiy[k].f1.set_size(1, 10);
65         ccoordix[k].f1.set_size(1, 10);
66         for (nx = 0; nx < 10; nx++) {
67             ccoordiy[k].f1[nx] = coordiy[k].f1[nx];
68             ccoordix[k].f1[nx] = coordix[k].f1[nx];
69         }
70     }
71     for (nx = 0; nx < 6000000; nx++) {
72         unsigned long long u;
73         u = binaryMap[nx];
74         b_binaryMap[nx] = binaryMap[nx];
75         if (binaryMap[nx] < 230ULL) {
76             u = OULL;
77             b_binaryMap[nx] = OULL;
78         }
79         if (u >= 230ULL) {
80             b_binaryMap[nx] = 255ULL;
81         }
82     }
83 }
84
85 //     se = offsetstrel('ball',5,3);
86 std::memcpy(&coveredMap[0], &b_binaryMap[0], 6000000U * sizeof(unsigned long
87 long));
88 imclose(coveredMap, b_binaryMap);
89 std::memcpy(&coveredMap[0], &b_binaryMap[0], 6000000U * sizeof(unsigned long
90 long));
91 imopen(coveredMap, b_binaryMap);
92
93 //     coveredMap = cat(3, binaryMap, binaryMap, binaryMap);
94 for (int robot = 0; robot < 5; robot++) {
95     if (robot + 1 > 3) {
```

```

96     ccoordix[robot].f1.set_size(1, 1);
97     ccoordix[robot].f1[0] = rtNaN;
98     ccoordiy[robot].f1.set_size(1, 1);
99     ccoordiy[robot].f1[0] = rtNaN;
100 } else {
101     x.set_size(ccoordix[robot].f1.size(0), ccoordix[robot].f1.size(1));
102     nx = ccoordix[robot].f1.size(0) * ccoordix[robot].f1.size(1);
103     for (k = 0; k < nx; k++) {
104         x[k] = ccoordix[robot].f1[k];
105     }
106
107     nx = ccoordix[robot].f1.size(1);
108     for (k = 0; k < nx; k++) {
109         x[k] = rt_roundd_snf(x[k]);
110     }
111
112     ccoordix[robot].f1.set_size(1, x.size(1));
113     nx = x.size(0) * x.size(1);
114     for (k = 0; k < nx; k++) {
115         ccoordix[robot].f1[k] = x[k];
116     }
117
118     x.set_size(ccordiy[robot].f1.size(0), ccoordiy[robot].f1.size(1));
119     nx = ccoordiy[robot].f1.size(0) * ccoordiy[robot].f1.size(1);
120     for (k = 0; k < nx; k++) {
121         x[k] = ccoordiy[robot].f1[k];
122     }
123
124     nx = ccoordiy[robot].f1.size(1);
125     for (k = 0; k < nx; k++) {
126         x[k] = rt_roundd_snf(x[k]);
127     }
128
129     ccoordiy[robot].f1.set_size(1, x.size(1));
130     nx = x.size(0) * x.size(1);
131     for (k = 0; k < nx; k++) {
132         ccoordiy[robot].f1[k] = x[k];
133     }
134 }
135 }
136
137 // imshow(binaryMap);
138 sensorParam.fov = 1.2217304763960306;
139 sensorParam.dFOV = 0.0043633231299858239;
140 sensorParam.radius = 300.0;
141 sensorParam.dvec = 15.0;
142 sensorParam.scolor = 100.0;
143 sensorParam.dc = 1.0;
144 sensorParam.Rmin = 75.0;
145 sensorParam.size[0] = 3000.0;
146 sensorParam.size[1] = 2000.0;
147 sensorParam.cs = 49.0;
148 sensorParam.mindis = 150.0;
149 sensorParam.H = 3000.0;
150 sensorParam.W = 2000.0;
151
152 // %% Léjöi řù
153 KWcalcMapCoverage(b_binaryMap, ccoordix, ccoordiy, coveredMap);
154
155 // coverager Mapbinary Map3DoFyüs t
156 // kw_imshow(ccoordix, ccoordiy);
157 // kw_dispCoveragePercent();
158 // % % % % % % % % % % % % % % % % % % % % % % % % % % % %
159 cluster[0] = ccoordix[0];
160 cell_index[0] = ccoordiy[0];
161 cluster[1] = ccoordix[1];
162 cell_index[1] = ccoordiy[1];
163 cluster[2] = ccoordix[2];
164 cell_index[2] = ccoordiy[2];
165 num = b_kw_cellDecomposition(coveredMap, cluster, cell_index);
166 for (k = 0; k < 6; k++) {
167     prev_cell[k] = 0.0;
168 }
169
170 int exitg1;
171 do {
172     exitg1 = 0;
173     KWcontactCell(b_binaryMap, cluster, num, ccoordix, ccoordiy, &nearClusterNum,
174                     &b_robot, &node, point, &phase);
175     if (nearClusterNum == 0.0) {
176         exitg1 = 1;
177     } else {
178         boolean_T y;
179         boolean_T exitg2;
180         b_x[0] = (nearClusterNum != prev_cell[0]);
181         b_x[1] = (b_robot != prev_cell[1]);
182         b_x[2] = (node != prev_cell[2]);
183

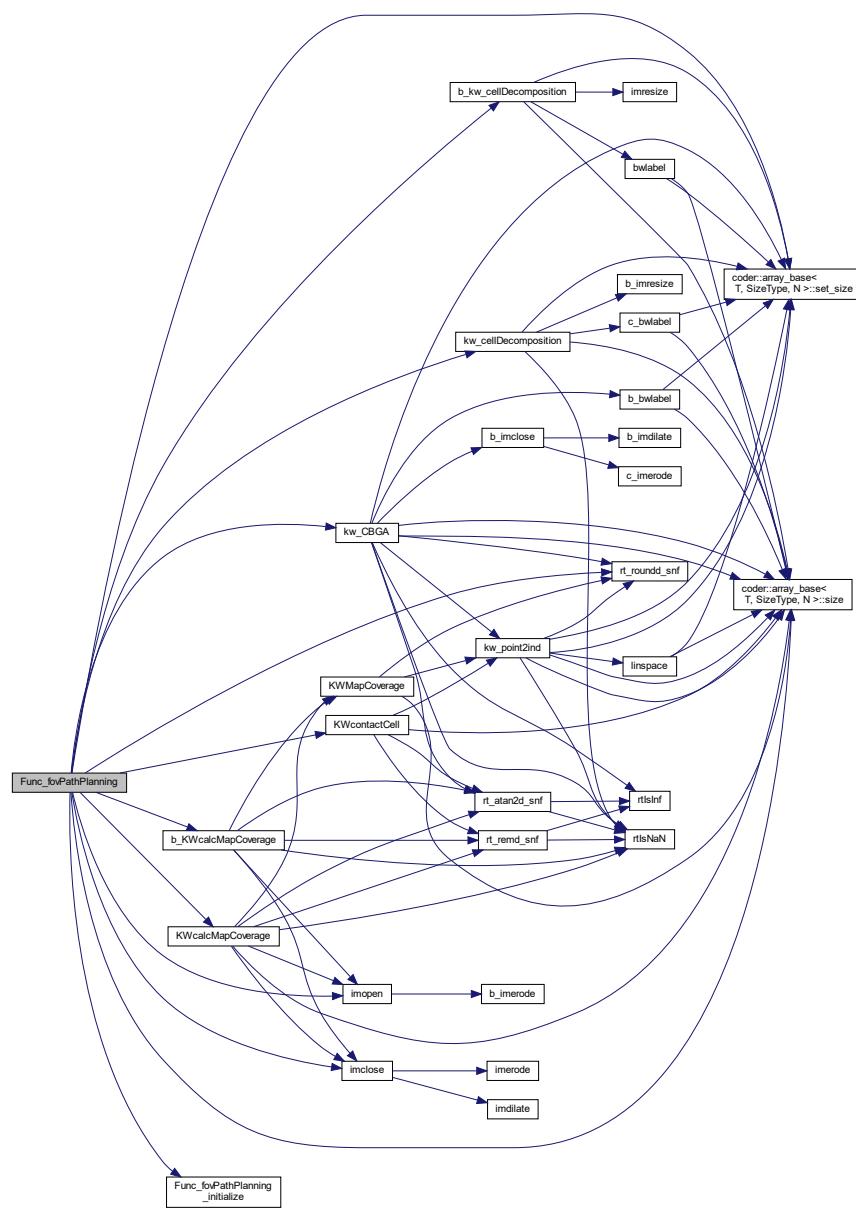
```

```

184     b_x[3] = (point[0] != prev_cell[3]);
185     b_x[4] = (point[1] != prev_cell[4]);
186     b_x[5] = (phase != prev_cell[5]);
187     y = false;
188     k = 0;
189     exitg2 = false;
190     while ((!exitg2) && (k < 6)) {
191         if (!b_x[k]) {
192             k++;
193         } else {
194             y = true;
195             exitg2 = true;
196         }
197     }
198
199     if (!y) {
200         double point_idx_1;
201         num = point[0] + 150.0 * std::cos(phase);
202         point_idx_1 = point[1] + 150.0 * std::sin(phase);
203         point[0] = num;
204         point[1] = point_idx_1;
205     }
206
207     prev_cell[0] = nearClusterNum;
208     prev_cell[1] = b_robot;
209     prev_cell[2] = node;
210     prev_cell[3] = point[0];
211     prev_cell[4] = point[1];
212     prev_cell[5] = phase;
213     kw_CBGA(&sensorParam, coveredMap, b_binaryMap, point, phase, unusedExpr);
214
215     // coordix{robot} = [ccoordix{robot}(1:node-1), path(1,1:8),
216     ccoordix{robot}(node+1:end)];
217     // coordiy{robot} = [ccoordiy{robot}(1:node-1), path(2,1:8),
218     ccoordiy{robot}(node+1:end)];
219     kw_cellDecomposition(coveredMap, coordix, coordiy, cluster, cell_index,
220                           &num);
221
222     // function [cells, cell_index, numCluster] =
223     kw_cellDecomposition(sensorParam, coveredMap, coordix, coordiy, showing_flag)
224 }
225 // new covered calculate
226 // for robot = 1:5
227 //     if isnan(coordix{robot})
228 //         continue;
229 //     end
230 //     fx = coordix{robot};
231 //     fy = coordiy{robot};
232 //     ix = icoordix{robot};
233 //     iy = icoordiy{robot};
234 //     coordix{robot} = [fx(:,1:end), fliplr(ix(:,1:end-1))];
235 //     coordiy{robot} = [fy(:,1:end), fliplr(iy(:,1:end-1))];
236 //     end
237 b_KWcalcMapCoverage(b_binaryMap, &sensorParam, coordix, coordiy, coveredMap);
238
239 // coverager Mapbinary Map3DoFýüs t
240 //     function [coveredMap]= KWcalcMapCoverage(binaryMap,sensorParam,coordix, coordiy,
241 //     show_flag)
242 //         [robotPoses, coorditheta] = kw_imshow(coordix, coordiy);
243 //         kw_dispCoveragePercent();
244 //         [coordix, coordiy, coorditheta, robotPoses] = KWdataPreProcessing2(coordix, coordiy,
245 //         coorditheta, robotPoses);
246 //         save('PrevkwResults.mat', 'coordix', 'coordiy', 'coorditheta', 'robotPoses');
247 }

```

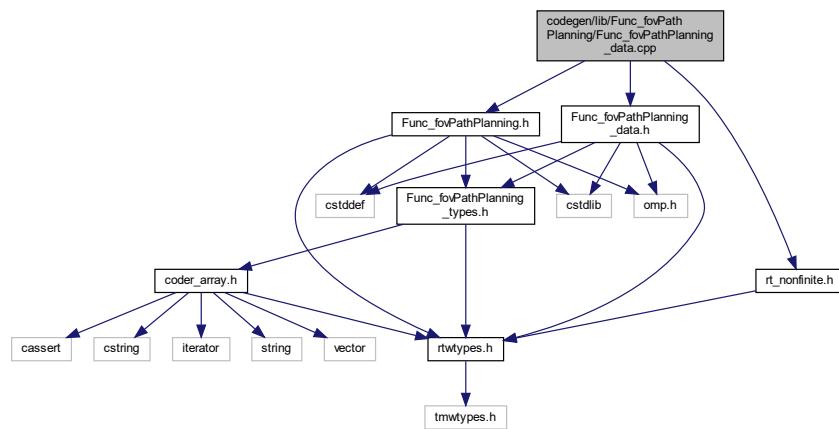
Here is the call graph for this function:



7.12 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_← data.cpp File Reference

```
#include "Func_fovPathPlanning_data.h"
#include "Func_fovPathPlanning.h"
#include "rt_nonfinite.h"
```

Include dependency graph for Func_fovPathPlanning_data.cpp:



Variables

- `omp_nest_lock_t emlrtNestLockGlobal`
- `const boolean_T bv [25]`
- `const boolean_T bv1 [25]`
- `const boolean_T bv2 [121]`
- `const boolean_T bv3 [121]`
- `boolean_T isInitialized_Func_fovPathPlanning = false`

7.12.1 Variable Documentation

7.12.1.1 bv

```
const boolean_T bv[25]
```

Initial value:

```
= { true, false, false, false, false, false, true, false,
  false, false, false, false, true, false, false, false, false, true,
  false, false, false, false, false, true }
```

Definition at line 19 of file Func_fovPathPlanning_data.cpp.

7.12.1.2 bv1

```
const boolean_T bv1[25]
```

Initial value:

```
= { false, false, false, false, true, false, false,
  false, true, false, false, true, false, false, false, true, false,
  false, false, true, false, false, false }
```

Definition at line 23 of file Func_fovPathPlanning_data.cpp.

7.12.1.3 bv2

```
const boolean_T bv2[121]
```

Initial value:

```
= { true, false, false, false, false, false,
  false, false, false, false, true, false, false, false, false, false,
  false, false, false, false, false, true, false, false, false, false,
  false, false, false, false, false, false, false, true, false, false,
  false, false, false, false, false, false, false, false, true, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  true, false, false, false, false, false, false, false, false, false,
  false, true, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, true,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, true, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, true, false, false, false, false, false, false, false,
  false, false, false, true }
```

Definition at line 27 of file Func_fovPathPlanning_data.cpp.

7.12.1.4 bv3

```
const boolean_T bv3[121]
```

Initial value:

```
= { false, false, false, false, false, false,
  false, false, false, true, false, false, false, false, false, false,
  false, false, true, false, false, false, false, false, false, false,
  false, true, false, false, false, false, false, false, false, false,
  true, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, true,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, true,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false,
  false, false, false, false, false, false, false, false, false, false }
```

Definition at line 40 of file Func_fovPathPlanning_data.cpp.

7.12.1.5 emlrtNestLockGlobal

```
omp_nest_lock_t emlrtNestLockGlobal
```

Definition at line 18 of file Func_fovPathPlanning_data.cpp.

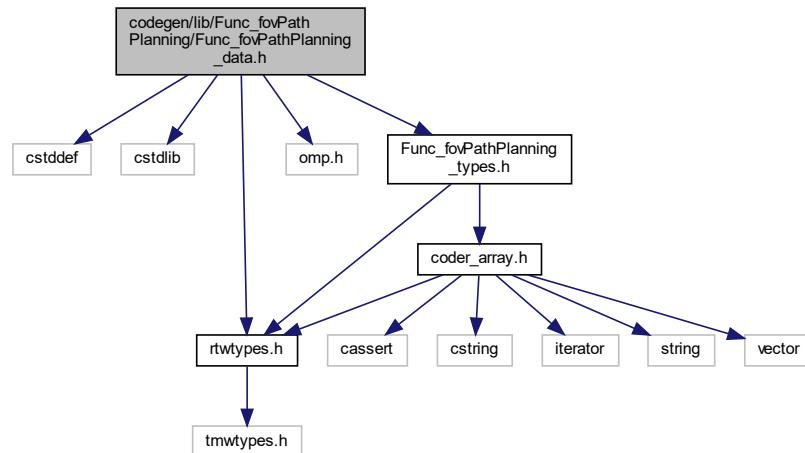
7.12.1.6 isInitialized_Func_fovPathPlanning

```
boolean_T isInitialized_Func_fovPathPlanning = false
```

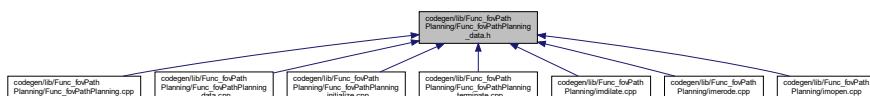
Definition at line 53 of file Func_fovPathPlanning_data.cpp.

7.13 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_data.h File Reference)

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for Func_fovPathPlanning_data.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Variables

- `omp_nest_lock_t emlRtNestLockGlobal`
- `const boolean_T bv [25]`
- `const boolean_T bv1 [25]`
- `const boolean_T bv2 [121]`
- `const boolean_T bv3 [121]`
- `boolean_T isInitialized_Func_fovPathPlanning`

7.13.1 Macro Definition Documentation

7.13.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 30 of file Func_fovPathPlanning_data.h.

7.13.2 Variable Documentation

7.13.2.1 bv

```
const boolean_T bv[25] [extern]
```

Definition at line 19 of file Func_fovPathPlanning_data.cpp.

7.13.2.2 bv1

```
const boolean_T bv1[25] [extern]
```

Definition at line 23 of file Func_fovPathPlanning_data.cpp.

7.13.2.3 bv2

```
const boolean_T bv2[121] [extern]
```

Definition at line 27 of file Func_fovPathPlanning_data.cpp.

7.13.2.4 bv3

```
const boolean_T bv3[121] [extern]
```

Definition at line 40 of file Func_fovPathPlanning_data.cpp.

7.13.2.5 emlrtNestLockGlobal

```
omp_nest_lock_t emlrtNestLockGlobal [extern]
```

Definition at line 18 of file Func_fovPathPlanning_data.cpp.

7.13.2.6 isInitialized_Func_fovPathPlanning

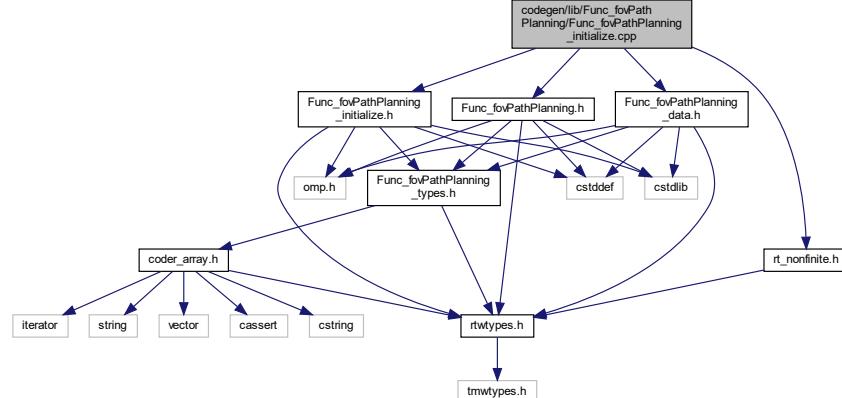
```
boolean_T isInitialized_Func_fovPathPlanning [extern]
```

Definition at line 53 of file Func_fovPathPlanning_data.cpp.

7.14 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_initialize.cpp) File Reference

```
#include "Func_fovPathPlanning_initialize.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_data.h"
#include "rt_nonfinite.h"
```

Include dependency graph for Func_fovPathPlanning_initialize.cpp:



Functions

- void [Func_fovPathPlanning_initialize \(\)](#)

7.14.1 Function Documentation

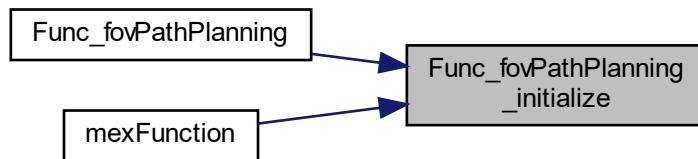
7.14.1.1 Func_fovPathPlanning_initialize()

```
void Func_fovPathPlanning_initialize (
    void )
```

Definition at line 19 of file Func_fovPathPlanning_initialize.cpp.

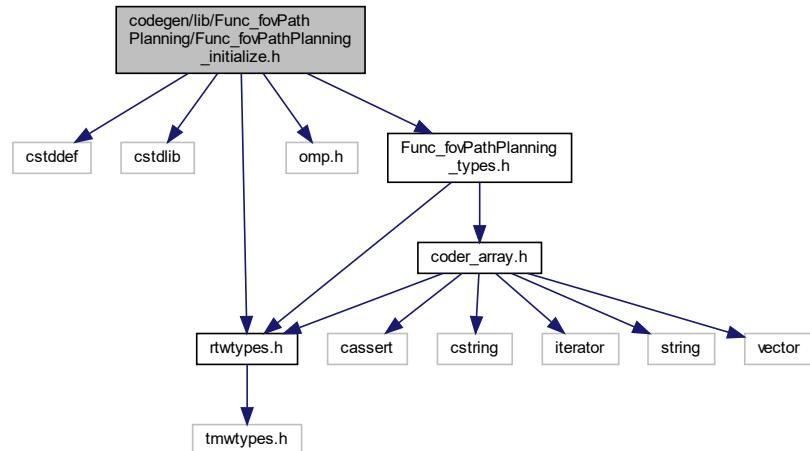
```
20 {
21     rt_InitInfAndNaN();
22     omp_init_nest_lock(&emlrtNestLockGlobal);
23     isInitialized_Func_fovPathPlanning = true;
24 }
```

Here is the caller graph for this function:

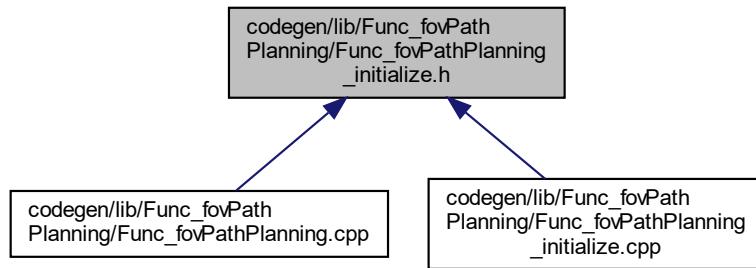


7.15 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_initialize.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for Func_fovPathPlanning_initialize.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void Func_fovPathPlanning_initialize ()`

7.15.1 Macro Definition Documentation

7.15.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file Func_fovPathPlanning_initialize.h.

7.15.2 Function Documentation

7.15.2.1 Func_fovPathPlanning_initialize()

```
void Func_fovPathPlanning_initialize ( )
```

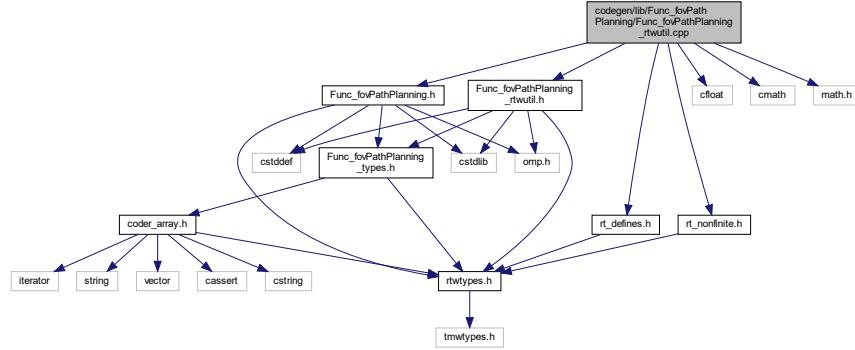
Definition at line 19 of file Func_fovPathPlanning_initialize.cpp.

```
20 {
21     rt_InitInfAndNaN();
22     omp_init_nest_lock(&emlrtNestLockGlobal);
23     isInitialized_Func_fovPathPlanning = true;
24 }
```

7.16 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_rtwutil.cpp File Reference

```
#include "Func_fovPathPlanning_rtwutil.h"
#include "Func_fovPathPlanning.h"
#include "rt_defines.h"
#include "rt_nonfinite.h"
#include <cfloat>
#include <cmath>
#include <math.h>
```

Include dependency graph for Func_fovPathPlanning_rtwutil.cpp:



Functions

- double `rt_atan2d_snf` (double u0, double u1)
- double `rt_remd_snf` (double u0, double u1)
- double `rt_rounnd_snf` (double u)

7.16.1 Function Documentation

7.16.1.1 `rt_atan2d_snf()`

```
double rt_atan2d_snf (
    double u0,
    double u1 )
```

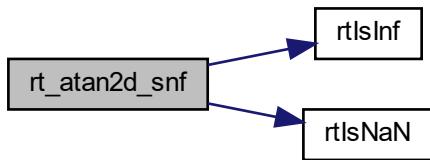
Definition at line 22 of file Func_fovPathPlanning_rtwutil.cpp.

```
23 {
24     double y;
25     if (rtIsNaN(u0) || rtIsNaN(u1)) {
26         y = rtNaN;
27     } else if (rtIsInf(u0) && rtIsInf(u1)) {
28         int b_u0;
29         int b_u1;
30         if (u0 > 0.0) {
31             b_u0 = 1;
32         } else {
33             b_u0 = -1;
```

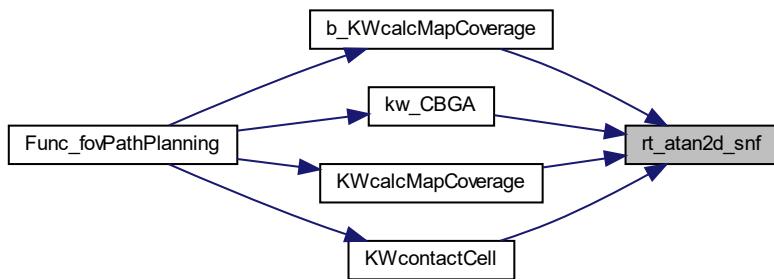
```

34     }
35
36     if (u1 > 0.0) {
37         b_u1 = 1;
38     } else {
39         b_u1 = -1;
40     }
41
42     y = atan2(static_cast<double>(b_u0), static_cast<double>(b_u1));
43 } else if (u1 == 0.0) {
44     if (u0 > 0.0) {
45         y = RT_PI / 2.0;
46     } else if (u0 < 0.0) {
47         y = -(RT_PI / 2.0);
48     } else {
49         y = 0.0;
50     }
51 } else {
52     y = atan2(u0, u1);
53 }
54
55 return y;
56 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.1.2 rt_remd_snf()

```

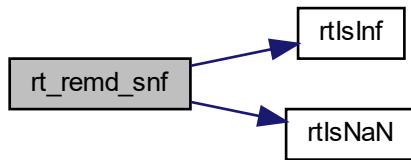
double rt_remd_snf (
    double u0,
    double u1 )
```

Definition at line 58 of file Func_fovPathPlanning_rtwutil.cpp.

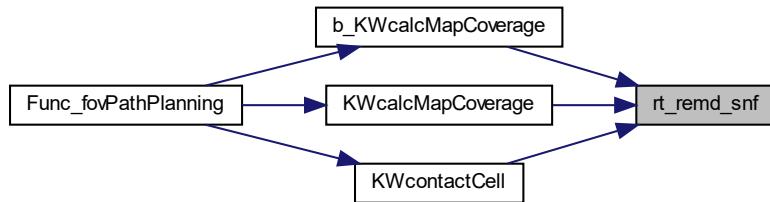
```

59 {
60     double y;
61     if (rtIsNaN(u0) || rtIsNaN(u1) || rtIsInf(u0)) {
62         y = rtNaN;
63     } else if (rtIsInf(u1)) {
64         y = u0;
65     } else {
66         double b_u1;
67         if (u1 < 0.0) {
68             b_u1 = std::ceil(u1);
69         } else {
70             b_u1 = std::floor(u1);
71         }
72
73         if ((u1 != 0.0) && (u1 != b_u1)) {
74             b_u1 = std::abs(u0 / u1);
75             if (!(std::abs(b_u1 - std::floor(b_u1 + 0.5)) > DBL_EPSILON * b_u1)) {
76                 y = 0.0 * u0;
77             } else {
78                 y = std::fmod(u0, u1);
79             }
80         } else {
81             y = std::fmod(u0, u1);
82         }
83     }
84
85     return y;
86 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



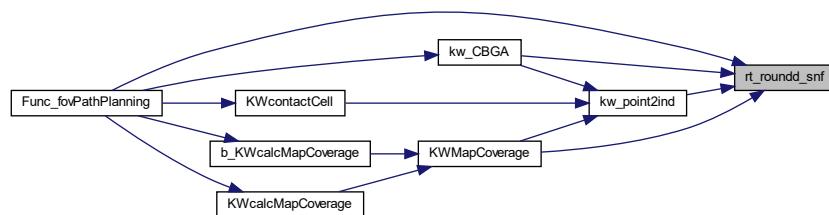
7.16.1.3 rt_roundd_snf()

```
double rt_roundd_snf (
    double u )
```

Definition at line 88 of file Func_fovPathPlanning_rtwutil.cpp.

```
89 {
90     double y;
91     if (std::abs(u) < 4.503599627370496E+15) {
92         if (u >= 0.5) {
93             y = std::floor(u + 0.5);
94         } else if (u > -0.5) {
95             y = u * 0.0;
96         } else {
97             y = std::ceil(u - 0.5);
98         }
99     } else {
100        y = u;
101    }
102
103    return y;
104 }
```

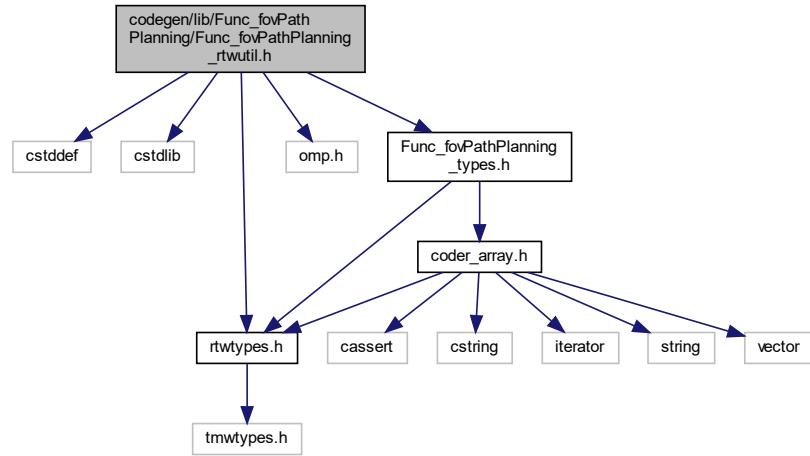
Here is the caller graph for this function:



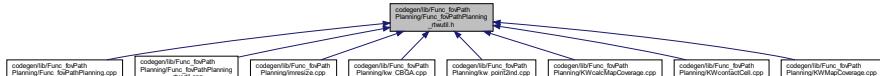
7.17 codegen/lib/Func_fovPathPlanning/Func_fovPathPlanning_rtwutil.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
```

Include dependency graph for Func_fovPathPlanning_rtwutil.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `double rt_atan2d_snf (double u0, double u1)`
- `double rt_remd_snf (double u0, double u1)`
- `double rt_roundd_snf (double u)`

7.17.1 Macro Definition Documentation

7.17.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file `Func_fovPathPlanning_rtwutil.h`.

7.17.2 Function Documentation

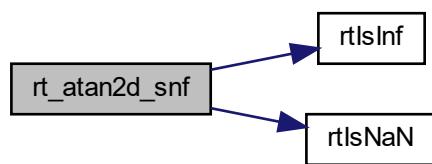
7.17.2.1 rt_atan2d_snf()

```
double rt_atan2d_snf (
    double u0,
    double u1 )
```

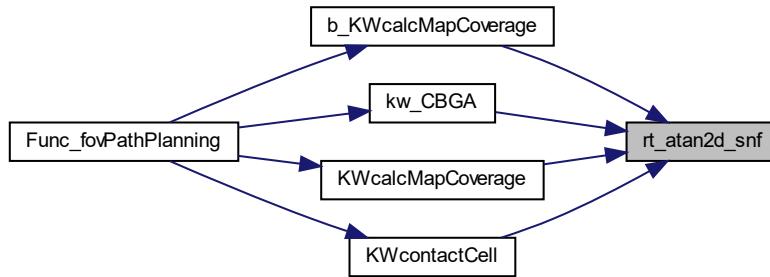
Definition at line 22 of file Func_fovPathPlanning_rtwutil.cpp.

```
23 {
24     double y;
25     if (rtIsNaN(u0) || rtIsNaN(u1)) {
26         y = rtNaN;
27     } else if (rtIsInf(u0) && rtIsInf(u1)) {
28         int b_u0;
29         int b_u1;
30         if (u0 > 0.0) {
31             b_u0 = 1;
32         } else {
33             b_u0 = -1;
34         }
35
36         if (u1 > 0.0) {
37             b_u1 = 1;
38         } else {
39             b_u1 = -1;
40         }
41
42         y = atan2(static_cast<double>(b_u0), static_cast<double>(b_u1));
43     } else if (u1 == 0.0) {
44         if (u0 > 0.0) {
45             y = RT_PI / 2.0;
46         } else if (u0 < 0.0) {
47             y = -(RT_PI / 2.0);
48         } else {
49             y = 0.0;
50         }
51     } else {
52         y = atan2(u0, u1);
53     }
54
55     return y;
56 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



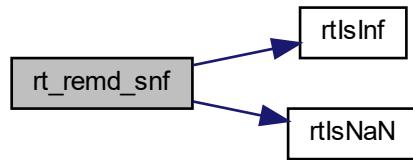
7.17.2.2 rt_remd_snf()

```
double rt_remd_snf (
    double u0,
    double u1 )
```

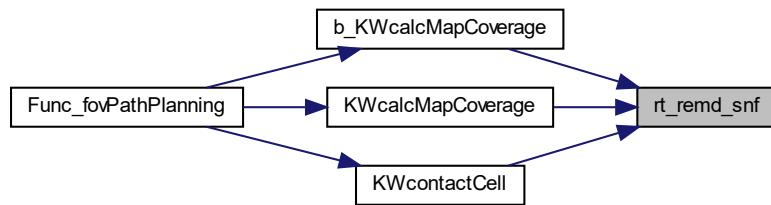
Definition at line 58 of file Func_fovPathPlanning_rtwutil.cpp.

```
59 {
60     double y;
61     if (rtIsNaN(u0) || rtIsNaN(u1) || rtIsInf(u0)) {
62         y = rtNaN;
63     } else if (rtIsInf(u1)) {
64         y = u0;
65     } else {
66         double b_u1;
67         if (u1 < 0.0) {
68             b_u1 = std::ceil(u1);
69         } else {
70             b_u1 = std::floor(u1);
71         }
72
73         if ((u1 != 0.0) && (u1 != b_u1)) {
74             b_u1 = std::abs(u0 / u1);
75             if (!(std::abs(b_u1 - std::floor(b_u1 + 0.5)) > DBL_EPSILON * b_u1)) {
76                 y = 0.0 * u0;
77             } else {
78                 y = std::fmod(u0, u1);
79             }
80         } else {
81             y = std::fmod(u0, u1);
82         }
83     }
84
85     return y;
86 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



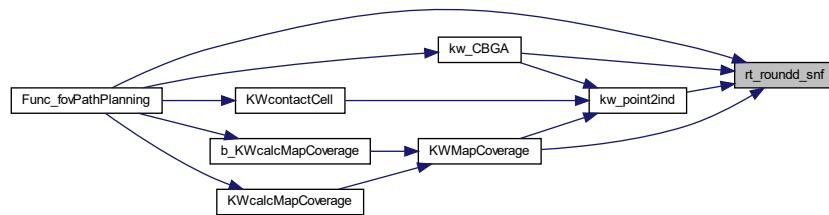
7.17.2.3 rt_roundd_snf()

```
double rt_roundd_snf (
    double u )
```

Definition at line 88 of file Func_fovPathPlanning_rtwutil.cpp.

```
89 {
90     double y;
91     if (std::abs(u) < 4.503599627370496E+15) {
92         if (u >= 0.5) {
93             y = std::floor(u + 0.5);
94         } else if (u > -0.5) {
95             y = u * 0.0;
96         } else {
97             y = std::ceil(u - 0.5);
98         }
99     } else {
100         y = u;
101     }
102
103     return y;
104 }
```

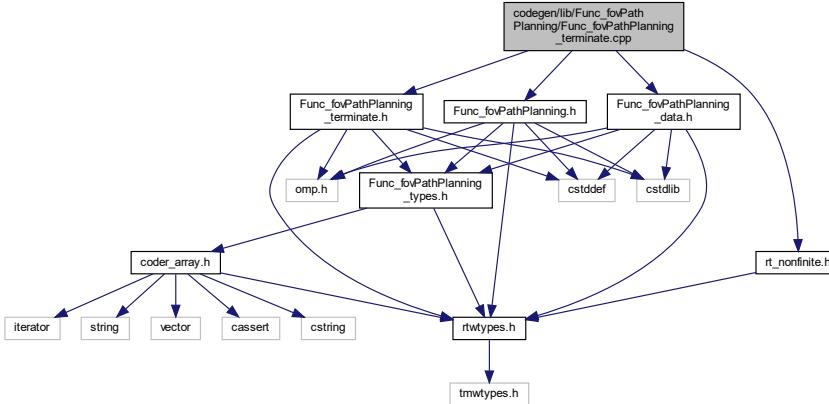
Here is the caller graph for this function:



7.18 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_terminate.cpp File Reference

```
#include "Func_fovPathPlanning_terminate.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_data.h"
#include "rt_nonfinite.h"
```

Include dependency graph for Func_fovPathPlanning_terminate.cpp:



Functions

- void [Func_fovPathPlanning_terminate \(\)](#)

7.18.1 Function Documentation

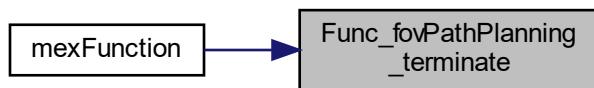
7.18.1.1 Func_fovPathPlanning_terminate()

```
void Func_fovPathPlanning_terminate (
    void )
```

Definition at line 19 of file Func_fovPathPlanning_terminate.cpp.

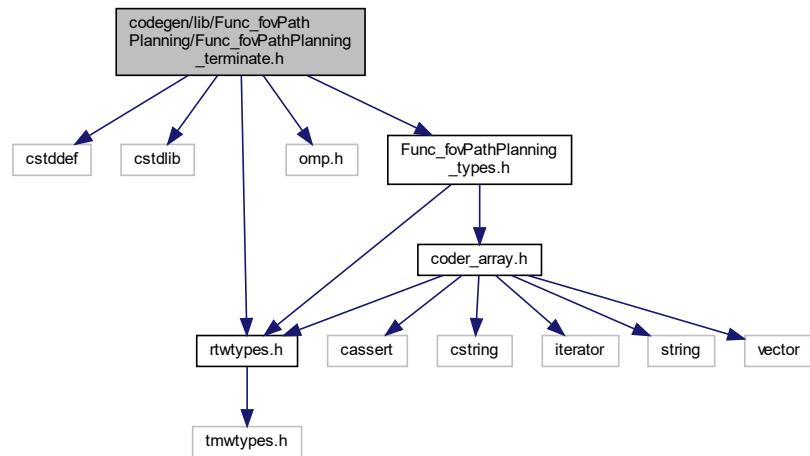
```
20 {
21     omp_destroy_nest_lock(&emlrtNestLockGlobal);
22     isInitialized_Func_fovPathPlanning = false;
23 }
```

Here is the caller graph for this function:

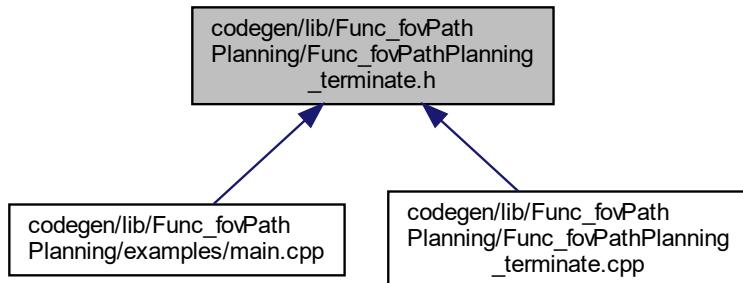


7.19 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_terminate.h) File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for Func_fovPathPlanning_terminate.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void Func_fovPathPlanning_terminate ()`

7.19.1 Macro Definition Documentation

7.19.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file `Func_fovPathPlanning_terminate.h`.

7.19.2 Function Documentation

7.19.2.1 Func_fovPathPlanning_terminate()

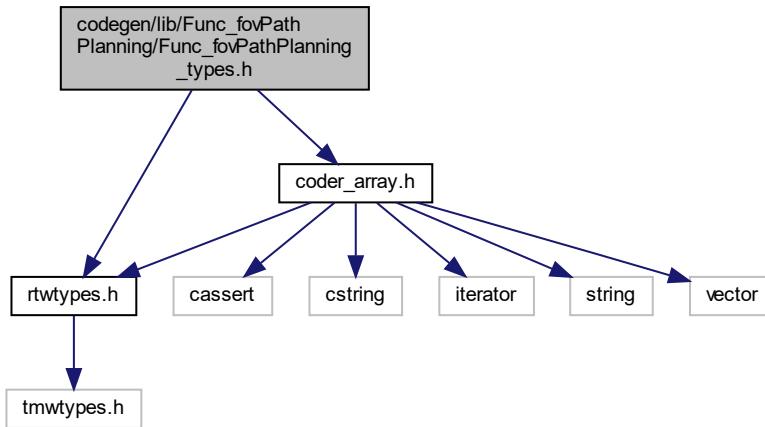
```
void Func_fovPathPlanning_terminate ( )
```

Definition at line 19 of file `Func_fovPathPlanning_terminate.cpp`.

```
20 {  
21     omp_destroy_nest_lock(&emlrtNestLockGlobal);  
22     isInitialized_Func_fovPathPlanning = false;  
23 }
```

7.20 codegen/lib(Func_fovPathPlanning/Func_fovPathPlanning_types.h File Reference

```
#include "rtwtypes.h"
#include "coder_array.h"
Include dependency graph for Func_fovPathPlanning_types.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct `cell_wrap_0`
- struct `cell_wrap_1`
- struct `struct_T`

Macros

- #define `MAX_THREADS` `omp_get_max_threads()`

7.20.1 Macro Definition Documentation

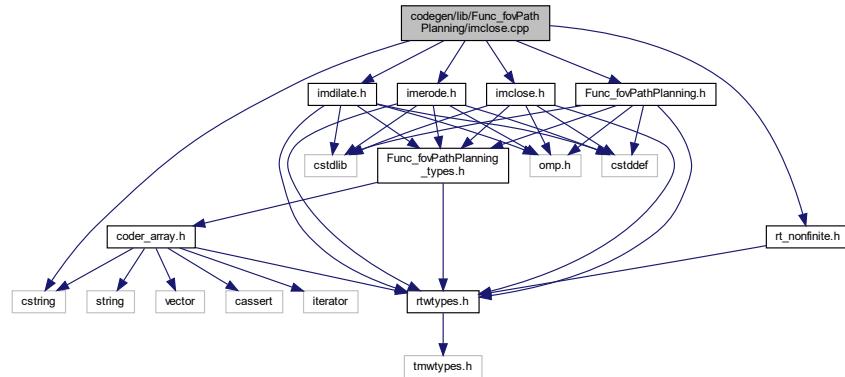
7.20.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 52 of file `Func_fovPathPlanning_types.h`.

7.21 codegen/lib/Func_fovPathPlanning/imclose.cpp File Reference

```
#include "imclose.h"
#include "Func_fovPathPlanning.h"
#include "imdilate.h"
#include "imerode.h"
#include "rt_nonfinite.h"
#include <cstring>
Include dependency graph for imclose.cpp:
```



Functions

- void [b_imclose](#) (const unsigned long long A[6000000], unsigned long long B[6000000])
- void [imclose](#) (const unsigned long long A[6000000], unsigned long long B[6000000])

7.21.1 Function Documentation

7.21.1.1 b_imclose()

```
void b_imclose (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

Definition at line 21 of file imclose.cpp.

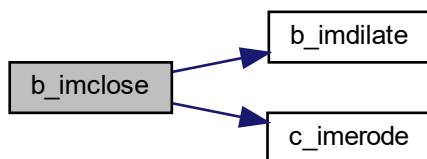
```
22 {
23     int j;
24     int i;
25     static unsigned long long Ap[6201600];
26     static unsigned long long uv[6201600];
27     for (j = 0; j < 20; j++) {
28         for (i = 0; i < 3040; i++) {
29             Ap[i + 3040 * j] = 0ULL;
30             Ap[i + 3040 * (j + 2020)] = 0ULL;
31         }
32     }
33
34     for (j = 0; j < 2000; j++) {
35         for (i = 0; i < 20; i++) {
36             int Ap_tmp;
```

```

37     Ap_tmp = i + 3040 * (j + 20);
38     Ap[Ap_tmp] = 0ULL;
39     Ap[Ap_tmp + 3020] = 0ULL;
40 }
41
42 std::memcpy(&Ap[j * 3040 + 60820], &A[j * 3000], 3000U * sizeof(unsigned
43     long long));
44 }
45
46 b_imdilate(Ap, uv);
47 c_imerode(uv, Ap);
48 for (j = 0; j < 2000; j++) {
49     std::memcpy(&B[j * 3000], &Ap[j * 3040 + 60820], 3000U * sizeof(unsigned
50     long long));
51 }
52 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.1.2 imclose()

```

void imclose (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )

```

Definition at line 54 of file imclose.cpp.

```

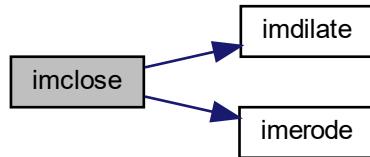
55 {
56     int j;
57     int i;
58     static unsigned long long Ap[6100400];
59     static unsigned long long uv[6100400];
60     for (j = 0; j < 10; j++) {
61         for (i = 0; i < 3020; i++) {
62             Ap[i + 3020 * j] = 0ULL;
63             Ap[i + 3020 * (j + 2010)] = 0ULL;
64         }

```

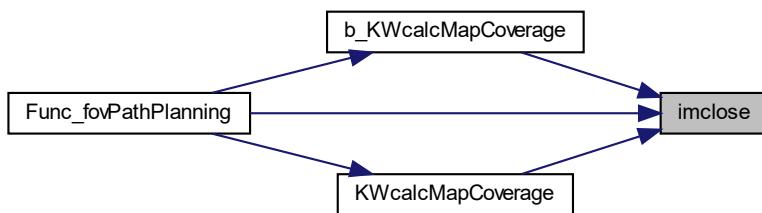
```

65     }
66
67     for (j = 0; j < 2000; j++) {
68         for (i = 0; i < 10; i++) {
69             int Ap_tmp;
70             Ap_tmp = i + 3020 * (j + 10);
71             Ap[Ap_tmp] = 0ULL;
72             Ap[Ap_tmp + 3010] = 0ULL;
73         }
74
75         std::memcpy(&Ap[j * 3020 + 30210], &A[j * 3000], 3000U * sizeof(unsigned
76             long long));
77     }
78
79     imdilate(Ap, uv);
80     imerode(uv, Ap);
81     for (j = 0; j < 2000; j++) {
82         std::memcpy(&B[j * 3000], &Ap[j * 3020 + 30210], 3000U * sizeof(unsigned
83             long long));
84     }
85 }
```

Here is the call graph for this function:



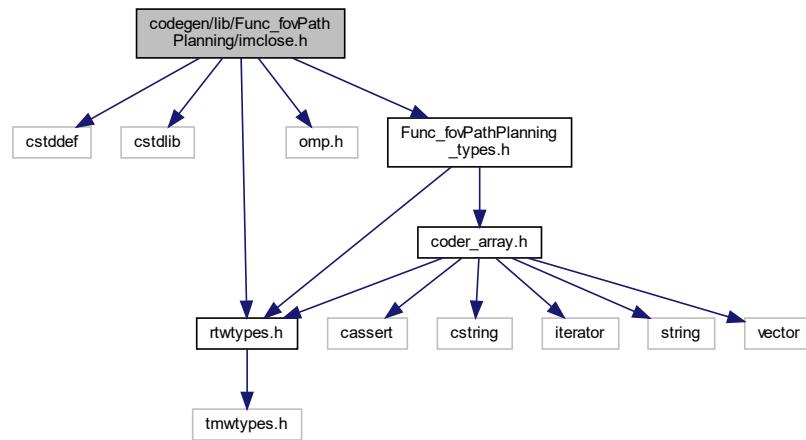
Here is the caller graph for this function:



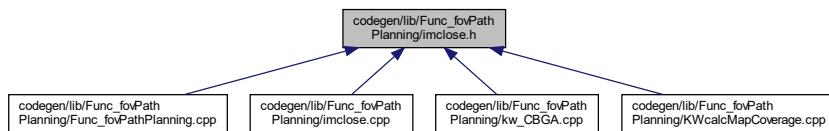
7.22 codegen/lib/Func_fovPathPlanning/imclose.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
```

```
#include "Func_fovPathPlanning_types.h"
Include dependency graph for imclose.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void b_imclose (const unsigned long long A[6000000], unsigned long long B[6000000])`
- `void imclose (const unsigned long long A[6000000], unsigned long long B[6000000])`

7.22.1 Macro Definition Documentation

7.22.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file imclose.h.

7.22.2 Function Documentation

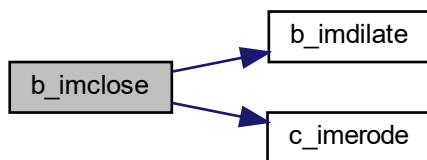
7.22.2.1 b_imclose()

```
void b_imclose (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

Definition at line 21 of file imclose.cpp.

```
22 {
23     int j;
24     int i;
25     static unsigned long long Ap[6201600];
26     static unsigned long long uv[6201600];
27     for (j = 0; j < 20; j++) {
28         for (i = 0; i < 3040; i++) {
29             Ap[i + 3040 * j] = OULL;
30             Ap[i + 3040 * (j + 2020)] = OULL;
31         }
32     }
33
34     for (j = 0; j < 2000; j++) {
35         for (i = 0; i < 20; i++) {
36             int Ap_tmp;
37             Ap_tmp = i + 3040 * (j + 20);
38             Ap[Ap_tmp] = OULL;
39             Ap[Ap_tmp + 3020] = OULL;
40         }
41
42     std::memcpy(&Ap[j * 3040 + 60820], &A[j * 3000], 3000U * sizeof(unsigned
43         long long));
44 }
45
46 b_imdilate(Ap, uv);
47 c_imerode(uv, Ap);
48 for (j = 0; j < 2000; j++) {
49     std::memcpy(&B[j * 3000], &Ap[j * 3040 + 60820], 3000U * sizeof(unsigned
50         long long));
51 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



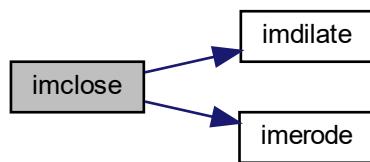
7.22.2.2 imclose()

```
void imclose (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

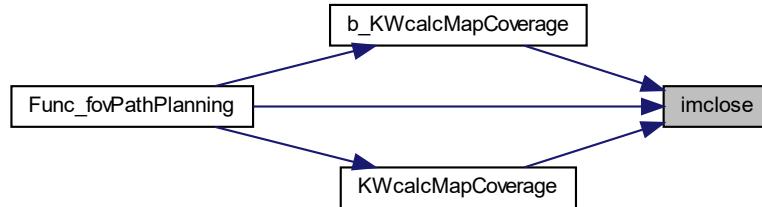
Definition at line 54 of file imclose.cpp.

```
55 {
56     int j;
57     int i;
58     static unsigned long long Ap[6100400];
59     static unsigned long long uv[6100400];
60     for (j = 0; j < 10; j++) {
61         for (i = 0; i < 3020; i++) {
62             Ap[i + 3020 * j] = OULL;
63             Ap[i + 3020 * (j + 2010)] = OULL;
64         }
65     }
66
67     for (j = 0; j < 2000; j++) {
68         for (i = 0; i < 10; i++) {
69             int Ap_tmp;
70             Ap_tmp = i + 3020 * (j + 10);
71             Ap[Ap_tmp] = OULL;
72             Ap[Ap_tmp + 3010] = OULL;
73         }
74
75         std::memcpy(&Ap[j * 3020 + 30210], &A[j * 3000], 3000U * sizeof(unsigned
76             long long));
77     }
78
79     imdilate(Ap, uv);
80     imerode(uv, Ap);
81     for (j = 0; j < 2000; j++) {
82         std::memcpy(&B[j * 3000], &Ap[j * 3020 + 30210], 3000U * sizeof(unsigned
83             long long));
84     }
85 }
```

Here is the call graph for this function:



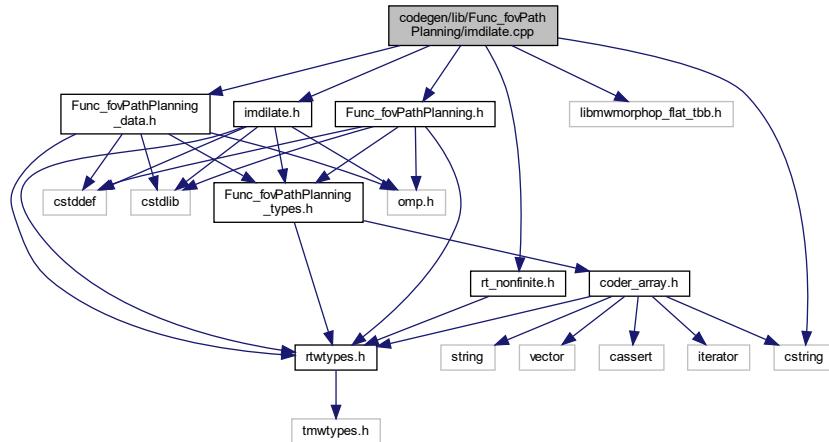
Here is the caller graph for this function:



7.23 codegen/lib/Func_fovPathPlanning/imdilate.cpp File Reference

```

#include "imdilate.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_data.h"
#include "libmwmorphop_flat_tbb.h"
#include "rt_nonfinite.h"
#include <cstring>
Include dependency graph for imdilate.cpp:
  
```



Functions

- void [b_imdilate](#) (const unsigned long long A[6201600], unsigned long long B[6201600])
- void [imdilate](#) (const unsigned long long A[6100400], unsigned long long B[6100400])

7.23.1 Function Documentation

7.23.1.1 b_imdilate()

```
void b_imdilate (
    const unsigned long long A[6201600],
    unsigned long long B[6201600] )
```

Definition at line 21 of file imdilate.cpp.

```
23 {
24     int i;
25     static unsigned long long Apadpre[6298481];
26     static unsigned long long Apad[6396084];
27     double asizeT[2];
28     boolean_T nhood[15];
29     double nsizeT[2];
30     static unsigned long long Apadpack[6396084];
31     boolean_T b_nhood[5];
32     for (i = 0; i < 19; i++) {
33         std::memset(&Apadpre[i * 3059], 0, 3059U * sizeof(unsigned long long));
34     }
35
36     for (i = 0; i < 2040; i++) {
37         std::memset(&Apadpre[i * 3059 + 58121], 0, 19U * sizeof(unsigned long long));
38         std::memcpy(&Apadpre[i * 3059 + 58140], &A[i * 3040], 3040U * sizeof
39                     (unsigned long long));
40     }
41
42     for (i = 0; i < 19; i++) {
43         std::memset(&Apad[i * 3078 + 6337602], 0, 3078U * sizeof(unsigned long long));
44     }
45
46     for (i = 0; i < 2059; i++) {
47         std::memset(&Apad[i * 3078 + 3059], 0, 19U * sizeof(unsigned long long));
48         std::memcpy(&Apad[i * 3078], &Apadpre[i * 3059], 3059U * sizeof(unsigned
49                     long long));
50     }
51
52     for (i = 0; i < 15; i++) {
53         nhood[i] = true;
54     }
55
56     asizeT[0] = 3078.0;
57     nsizeT[0] = 15.0;
58     asizeT[1] = 2078.0;
59     nsizeT[1] = 1.0;
60     dilate_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
61     asizeT[0] = 3078.0;
62     nsizeT[0] = 11.0;
63     asizeT[1] = 2078.0;
64     nsizeT[1] = 11.0;
65     dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv2, nsizeT, 2.0, Apad);
66     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
67     for (i = 0; i < 15; i++) {
68         nhood[i] = true;
69     }
70
71     asizeT[0] = 3078.0;
72     nsizeT[0] = 1.0;
73     asizeT[1] = 2078.0;
74     nsizeT[1] = 15.0;
75     dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
76     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
77     asizeT[0] = 3078.0;
78     nsizeT[0] = 11.0;
79     asizeT[1] = 2078.0;
80     nsizeT[1] = 11.0;
81     dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv3, nsizeT, 2.0, Apad);
82     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
83     for (i = 0; i < 5; i++) {
84         b_nhood[i] = true;
85     }
86
87     asizeT[0] = 3078.0;
88     nsizeT[0] = 1.0;
89     asizeT[1] = 2078.0;
90     nsizeT[1] = 5.0;
91     dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
92     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
93     for (i = 0; i < 5; i++) {
94         b_nhood[i] = true;
95     }
96
97     asizeT[0] = 3078.0;
98     nsizeT[0] = 5.0;
99     asizeT[1] = 2078.0;
```

```

100    nsizeT[1] = 1.0;
101    dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
102    for (i = 0; i < 2040; i++) {
103        std::memcpy(&B[i * 3040], &Apad[i * 3078 + 58501], 3040U * sizeof(unsigned
104        long long));
105    }
106 }

```

Here is the caller graph for this function:



7.23.1.2 imdilate()

```

void imdilate (
    const unsigned long long A[6100400],
    unsigned long long B[6100400] )

```

Definition at line 108 of file imdilate.cpp.

```

109 {
110     int i;
111     static unsigned long long Apadpre[6145841];
112     static unsigned long long Apad[6191444];
113     double asizeT[2];
114     boolean_T nhood[7];
115     double nsizeT[2];
116     static unsigned long long Apadpack[6191444];
117     boolean_T b_nhood[5];
118     for (i = 0; i < 9; i++) {
119         std::memset(&Apadpre[i * 3029], 0, 3029U * sizeof(unsigned long long));
120     }
121
122     for (i = 0; i < 2020; i++) {
123         std::memset(&Apadpre[i * 3029 + 27261], 0, 9U * sizeof(unsigned long long));
124         std::memcpy(&Apadpre[i * 3029 + 27270], &A[i * 3020], 3020U * sizeof
125             (unsigned long long));
126     }
127
128     for (i = 0; i < 9; i++) {
129         std::memset(&Apad[i * 3038 + 6164102], 0, 3038U * sizeof(unsigned long long));
130     }
131
132     for (i = 0; i < 2029; i++) {
133         std::memset(&Apad[i * 3038 + 3029], 0, 9U * sizeof(unsigned long long));
134         std::memcpy(&Apad[i * 3038], &Apadpre[i * 3029], 3029U * sizeof(unsigned
135             long long));
136     }
137
138     for (i = 0; i < 7; i++) {
139         nhood[i] = true;
140     }
141
142     asizeT[0] = 3038.0;
143     nsizeT[0] = 7.0;
144     asizeT[1] = 2038.0;
145     nsizeT[1] = 1.0;
146     dilate_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
147     asizeT[0] = 3038.0;
148     nsizeT[0] = 5.0;
149     asizeT[1] = 2038.0;
150     nsizeT[1] = 5.0;
151     dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
152     std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
153     for (i = 0; i < 7; i++) {
154         nhood[i] = true;

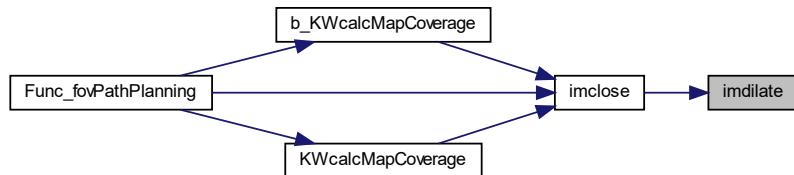
```

```

155 }
156 asizeT[0] = 3038.0;
158 nsizet[0] = 1.0;
159 asizeT[1] = 2038.0;
160 nsizet[1] = 7.0;
161 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizet, 2.0, Apad);
162 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
163 asizeT[0] = 3038.0;
164 nsizet[0] = 5.0;
165 asizeT[1] = 2038.0;
166 nsizet[1] = 5.0;
167 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, bvl, nsizet, 2.0, Apad);
168 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
169 for (i = 0; i < 5; i++) {
170     b_nhood[i] = true;
171 }
172 asizeT[0] = 3038.0;
174 nsizet[0] = 1.0;
175 asizeT[1] = 2038.0;
176 nsizet[1] = 5.0;
177 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizet, 2.0, Apad);
178 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
179 for (i = 0; i < 5; i++) {
180     b_nhood[i] = true;
181 }
182 asizeT[0] = 3038.0;
184 nsizet[0] = 5.0;
185 asizeT[1] = 2038.0;
186 nsizet[1] = 1.0;
187 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizet, 2.0, Apad);
188 for (i = 0; i < 2020; i++) {
189     std::memcpy(&B[i * 3020], &Apad[i * 3038 + 27351], 3020U * sizeof(unsigned
190         long long));
191 }
192 }

```

Here is the caller graph for this function:



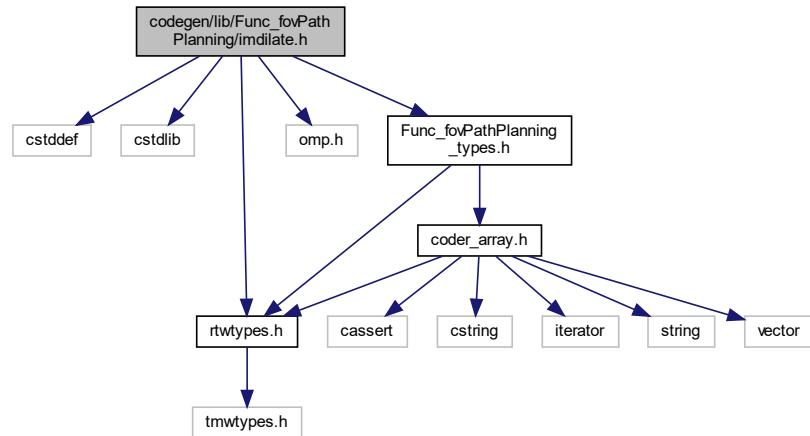
7.24 codegen/lib/Func_fovPathPlanning/imdilate.h File Reference

```

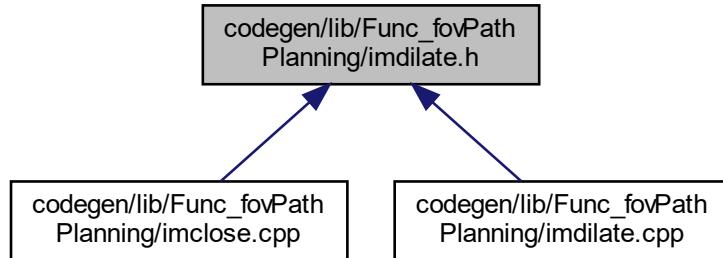
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"

```

Include dependency graph for imdilate.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void b_imdilate (const unsigned long long A[6201600], unsigned long long B[6201600])`
- `void imdilate (const unsigned long long A[6100400], unsigned long long B[6100400])`

7.24.1 Macro Definition Documentation

7.24.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file imdilate.h.

7.24.2 Function Documentation

7.24.2.1 b_imdilate()

```
void b_imdilate (
    const unsigned long long A[6201600],
    unsigned long long B[6201600] )
```

Definition at line 21 of file imdilate.cpp.

```
23 {
24     int i;
25     static unsigned long long Apadpre[6298481];
26     static unsigned long long Apad[6396084];
27     double asizeT[2];
28     boolean_T nhood[15];
29     double nsizet[2];
30     static unsigned long long Apadpack[6396084];
31     boolean_T b_nhood[5];
32     for (i = 0; i < 19; i++) {
33         std::memset(&Apadpre[i * 3059], 0, 3059U * sizeof(unsigned long long));
34     }
35
36     for (i = 0; i < 2040; i++) {
37         std::memset(&Apadpre[i * 3059 + 58121], 0, 19U * sizeof(unsigned long long));
38         std::memcpy(&Apadpre[i * 3059 + 58140], &A[i * 3040], 3040U * sizeof
39                     (unsigned long long));
40     }
41
42     for (i = 0; i < 19; i++) {
43         std::memset(&Apad[i * 3078 + 6337602], 0, 3078U * sizeof(unsigned long long));
44     }
45
46     for (i = 0; i < 2059; i++) {
47         std::memset(&Apad[i * 3078 + 3059], 0, 19U * sizeof(unsigned long long));
48         std::memcpy(&Apad[i * 3078], &Apadpre[i * 3059], 3059U * sizeof(unsigned
49                     long long));
50     }
51
52     for (i = 0; i < 15; i++) {
53         nhood[i] = true;
54     }
55
56     asizeT[0] = 3078.0;
57     nsizet[0] = 15.0;
58     asizeT[1] = 2078.0;
59     nsizet[1] = 1.0;
60     dilate_fflat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizet, 2.0, Apadpack);
61     asizeT[0] = 3078.0;
62     nsizet[0] = 11.0;
63     asizeT[1] = 2078.0;
64     nsizet[1] = 11.0;
65     dilate_fflat_uint64_tbb(Apadpack, asizet, 2.0, bv2, nsizet, 2.0, Apad);
66     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
67     for (i = 0; i < 15; i++) {
68         nhood[i] = true;
69     }
70
71     asizeT[0] = 3078.0;
72     nsizet[0] = 1.0;
73     asizeT[1] = 2078.0;
74     nsizet[1] = 15.0;
75     dilate_fflat_uint64_tbb(Apadpack, asizet, 2.0, nhood, nsizet, 2.0, Apad);
76     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
77     asizeT[0] = 3078.0;
78     nsizet[0] = 11.0;
```

```

79  asizeT[1] = 2078.0;
80  nsizet[1] = 11.0;
81  dilate_flat_uint64_tbb(Apadpack, asizet, 2.0, bv3, nsizet, 2.0, Apad);
82  std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
83  for (i = 0; i < 5; i++) {
84      b_nhood[i] = true;
85  }
86
87  asizeT[0] = 3078.0;
88  nsizet[0] = 1.0;
89  asizeT[1] = 2078.0;
90  nsizet[1] = 5.0;
91  dilate_flat_uint64_tbb(Apadpack, asizet, 2.0, b_nhood, nsizet, 2.0, Apad);
92  std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
93  for (i = 0; i < 5; i++) {
94      b_nhood[i] = true;
95  }
96
97  asizeT[0] = 3078.0;
98  nsizet[0] = 5.0;
99  asizeT[1] = 2078.0;
100 nsizet[1] = 1.0;
101 dilate_flat_uint64_tbb(Apadpack, asizet, 2.0, b_nhood, nsizet, 2.0, Apad);
102 for (i = 0; i < 2040; i++) {
103     std::memcpy(&B[i * 3040], &Apad[i * 3078 + 58501], 3040U * sizeof(unsigned
104         long long));
105 }
106 }
```

Here is the caller graph for this function:



7.24.2.2 imdilate()

```

void imdilate (
    const unsigned long long A[6100400],
    unsigned long long B[6100400] )
```

Definition at line 108 of file imdilate.cpp.

```

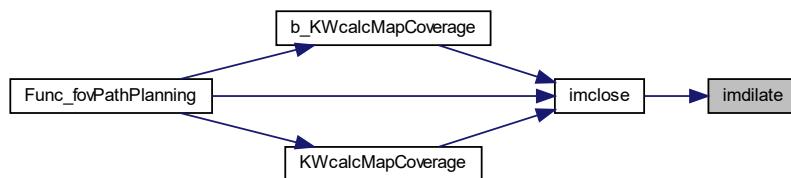
109 {
110     int i;
111     static unsigned long long Apadpre[6145841];
112     static unsigned long long Apad[6191444];
113     double asizeT[2];
114     boolean_T nhood[7];
115     double nsizet[2];
116     static unsigned long long Apadpack[6191444];
117     boolean_T b_nhood[5];
118     for (i = 0; i < 9; i++) {
119         std::memset(&Apadpre[i * 3029], 0, 3029U * sizeof(unsigned long long));
120     }
121
122     for (i = 0; i < 2020; i++) {
123         std::memset(&Apadpre[i * 3029 + 27261], 0, 9U * sizeof(unsigned long long));
124         std::memcpy(&Apadpre[i * 3029 + 27270], &A[i * 3020], 3020U * sizeof
125             (unsigned long long));
126     }
127
128     for (i = 0; i < 9; i++) {
129         std::memset(&Apad[i * 3038 + 6164102], 0, 3038U * sizeof(unsigned long long));
130     }
131
132     for (i = 0; i < 2029; i++) {
133         std::memset(&Apad[i * 3038 + 3029], 0, 9U * sizeof(unsigned long long));
```

```

134     std::memcpy(&Apad[i * 3038], &Apadpre[i * 3029], 3029U * sizeof(unsigned
135         long long));
136 }
137
138 for (i = 0; i < 7; i++) {
139     nhood[i] = true;
140 }
141
142 asizeT[0] = 3038.0;
143 nsizeT[0] = 7.0;
144 asizeT[1] = 2038.0;
145 nsizeT[1] = 1.0;
146 dilate_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
147 asizeT[0] = 3038.0;
148 nsizeT[0] = 5.0;
149 asizeT[1] = 2038.0;
150 nsizeT[1] = 5.0;
151 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
152 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
153 for (i = 0; i < 7; i++) {
154     nhood[i] = true;
155 }
156
157 asizeT[0] = 3038.0;
158 nsizeT[0] = 1.0;
159 asizeT[1] = 2038.0;
160 nsizeT[1] = 7.0;
161 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
162 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
163 asizeT[0] = 3038.0;
164 nsizeT[0] = 5.0;
165 asizeT[1] = 2038.0;
166 nsizeT[1] = 5.0;
167 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, bvl, nsizeT, 2.0, Apad);
168 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
169 for (i = 0; i < 5; i++) {
170     b_nhood[i] = true;
171 }
172
173 asizeT[0] = 3038.0;
174 nsizeT[0] = 1.0;
175 asizeT[1] = 2038.0;
176 nsizeT[1] = 5.0;
177 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
178 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
179 for (i = 0; i < 5; i++) {
180     b_nhood[i] = true;
181 }
182
183 asizeT[0] = 3038.0;
184 nsizeT[0] = 5.0;
185 asizeT[1] = 2038.0;
186 nsizeT[1] = 1.0;
187 dilate_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
188 for (i = 0; i < 2020; i++) {
189     std::memcpy(&B[i * 3020], &Apad[i * 3038 + 27351], 3020U * sizeof(unsigned
190         long long));
191 }
192 }

```

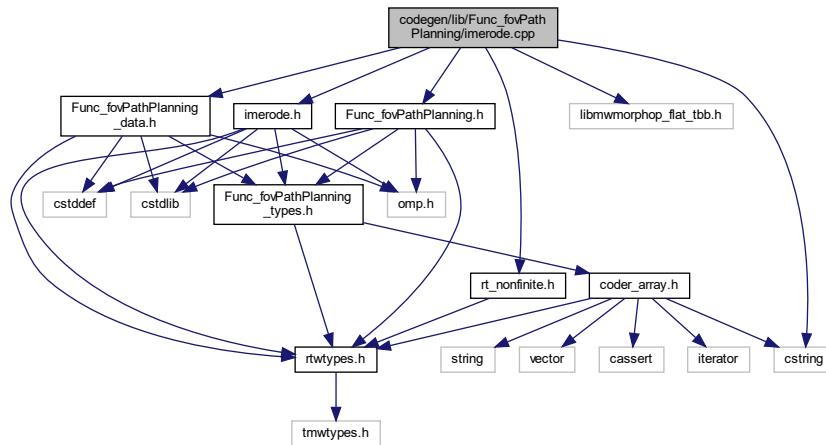
Here is the caller graph for this function:



7.25 codegen/lib/Func_fovPathPlanning/imerode.cpp File Reference

```
#include "imerode.h"
```

```
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_data.h"
#include "libmwmorphop_flat_tbb.h"
#include "rt_nonfinite.h"
#include <cstring>
Include dependency graph for imerode.cpp:
```



Functions

- void `b_imerode` (const unsigned long long A[6000000], unsigned long long B[6000000])
- void `c_imerode` (const unsigned long long A[6201600], unsigned long long B[6201600])
- void `imerode` (const unsigned long long A[6100400], unsigned long long B[6100400])

7.25.1 Function Documentation

7.25.1.1 `b_imerode()`

```
void b_imerode (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

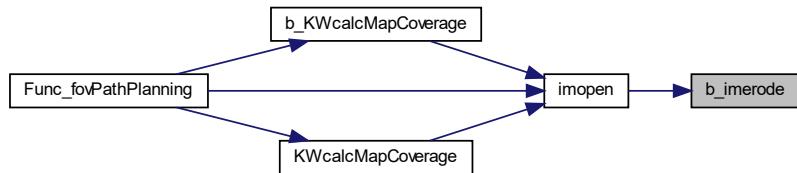
Definition at line 21 of file imerode.cpp.

```
22 {
23     int j;
24     int i;
25     static unsigned long long Apadpre[6045081];
26     static unsigned long long Apad[6090324];
27     double asizeT[2];
28     boolean_T nhood[7];
29     double nsizeT[2];
30     static unsigned long long Apadpack[6090324];
31     boolean_T b_nhhood[5];
32     for (j = 0; j < 9; j++) {
33         for (i = 0; i < 3009; i++) {
34             Apadpre[i + 3009 * j] = MAX_uint64_T;
35         }
36     }
```

```

37
38     for (j = 0; j < 2000; j++) {
39         for (i = 0; i < 9; i++) {
40             Apadpre[i + 3009 * (j + 9)] = MAX_uint64_T;
41         }
42
43         std::memcpy(&Apadpre[j * 3009 + 27090], &A[j * 3000], 3000U * sizeof
44             (unsigned long long));
45     }
46
47     for (j = 0; j < 9; j++) {
48         for (i = 0; i < 3018; i++) {
49             Apad[i + 3018 * (j + 2009)] = MAX_uint64_T;
50         }
51     }
52
53     for (j = 0; j < 2009; j++) {
54         for (i = 0; i < 9; i++) {
55             Apad[(i + 3018 * j) + 3009] = MAX_uint64_T;
56         }
57
58         std::memcpy(&Apad[j * 3018], &Apadpre[j * 3009], 3009U * sizeof(unsigned
59             long long));
60     }
61
62     for (i = 0; i < 7; i++) {
63         nhood[i] = true;
64     }
65
66     asizeT[0] = 3018.0;
67     nsizeT[0] = 7.0;
68     asizeT[1] = 2018.0;
69     nsizeT[1] = 1.0;
70     erode_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
71     asizeT[0] = 3018.0;
72     nsizeT[0] = 5.0;
73     asizeT[1] = 2018.0;
74     nsizeT[1] = 5.0;
75     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
76     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
77     for (i = 0; i < 7; i++) {
78         nhood[i] = true;
79     }
80
81     asizeT[0] = 3018.0;
82     nsizeT[0] = 1.0;
83     asizeT[1] = 2018.0;
84     nsizeT[1] = 7.0;
85     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
86     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
87     asizeT[0] = 3018.0;
88     nsizeT[0] = 5.0;
89     asizeT[1] = 2018.0;
90     nsizeT[1] = 5.0;
91     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bvl, nsizeT, 2.0, Apad);
92     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
93     for (j = 0; j < 5; j++) {
94         b_nhood[j] = true;
95     }
96
97     asizeT[0] = 3018.0;
98     nsizeT[0] = 1.0;
99     asizeT[1] = 2018.0;
100    nsizeT[1] = 5.0;
101    erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
102    std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
103    for (j = 0; j < 5; j++) {
104        b_nhood[j] = true;
105    }
106
107    asizeT[0] = 3018.0;
108    nsizeT[0] = 5.0;
109    asizeT[1] = 2018.0;
110    nsizeT[1] = 1.0;
111    erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
112    for (j = 0; j < 2000; j++) {
113        std::memcpy(&B[j * 3000], &Apad[j * 3018 + 27171], 3000U * sizeof(unsigned
114            long long));
115    }
116 }
```

Here is the caller graph for this function:



7.25.1.2 c_imerode()

```
void c_imerode (
    const unsigned long long A[6201600],
    unsigned long long B[6201600] )
```

Definition at line 118 of file imerode.cpp.

```

119 {
120     int j;
121     int i;
122     static unsigned long long Apadpre[6298481];
123     static unsigned long long Apad[6396084];
124     double asizeT[2];
125     boolean_T nhood[15];
126     double nsizeT[2];
127     static unsigned long long Apadpack[6396084];
128     boolean_T b_nhood[5];
129     for (j = 0; j < 19; j++) {
130         for (i = 0; i < 3059; i++) {
131             Apadpre[i + 3059 * j] = MAX_uint64_T;
132         }
133     }
134
135     for (j = 0; j < 2040; j++) {
136         for (i = 0; i < 19; i++) {
137             Apadpre[i + 3059 * (j + 19)] = MAX_uint64_T;
138         }
139
140         std::memcpy(&Apadpre[j * 3059 + 58140], &A[j * 3040], 3040U * sizeof
141             (unsigned long long));
142     }
143
144     for (j = 0; j < 19; j++) {
145         for (i = 0; i < 3078; i++) {
146             Apad[i + 3078 * (j + 2059)] = MAX_uint64_T;
147         }
148     }
149
150     for (j = 0; j < 2059; j++) {
151         for (i = 0; i < 19; i++) {
152             Apad[(i + 3078 * j) + 3059] = MAX_uint64_T;
153         }
154
155         std::memcpy(&Apad[j * 3078], &Apadpre[j * 3059], 3059U * sizeof(unsigned
156             long long));
157     }
158
159     for (i = 0; i < 15; i++) {
160         nhood[i] = true;
161     }
162
163     asizeT[0] = 3078.0;
164     nsizeT[0] = 15.0;
165     asizeT[1] = 2078.0;
166     nsizeT[1] = 1.0;
167     erode_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
168     asizeT[0] = 3078.0;
  
```

```

169     nsizeT[0] = 11.0;
170     asizeT[1] = 2078.0;
171     nsizeT[1] = 11.0;
172     erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bv2, nsizeT, 2.0, Apad);
173     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
174     for (i = 0; i < 15; i++) {
175         nhood[i] = true;
176     }
177
178     asizeT[0] = 3078.0;
179     nsizeT[0] = 1.0;
180     asizeT[1] = 2078.0;
181     nsizeT[1] = 15.0;
182     erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
183     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
184     asizeT[0] = 3078.0;
185     nsizeT[0] = 11.0;
186     asizeT[1] = 2078.0;
187     nsizeT[1] = 11.0;
188     erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bv3, nsizeT, 2.0, Apad);
189     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
190     for (j = 0; j < 5; j++) {
191         b_nhood[j] = true;
192     }
193
194     asizeT[0] = 3078.0;
195     nsizeT[0] = 1.0;
196     asizeT[1] = 2078.0;
197     nsizeT[1] = 5.0;
198     erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
199     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
200     for (j = 0; j < 5; j++) {
201         b_nhood[j] = true;
202     }
203
204     asizeT[0] = 3078.0;
205     nsizeT[0] = 5.0;
206     asizeT[1] = 2078.0;
207     nsizeT[1] = 1.0;
208     erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
209     for (j = 0; j < 2040; j++) {
210         std::memcpy(&B[j * 3040], &Apad[j * 3078 + 58501], 3040U * sizeof(unsigned
211             long long));
212     }
213 }
```

Here is the caller graph for this function:



7.25.1.3 imerode()

```

void imerode (
    const unsigned long long A[6100400],
    unsigned long long B[6100400] )
```

Definition at line 215 of file imerode.cpp.

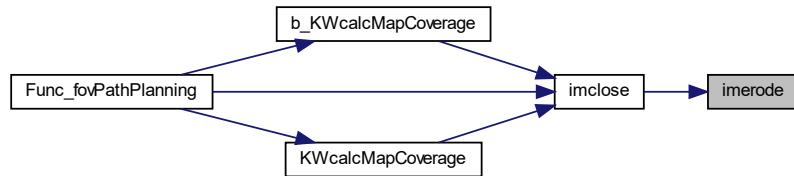
```

216 {
217     int j;
218     int i;
219     static unsigned long long Apadpre[6145841];
220     static unsigned long long Apad[6191444];
221     double asizeT[2];
222     boolean_T nhood[7];
223     double nsizeT[2];
```

```

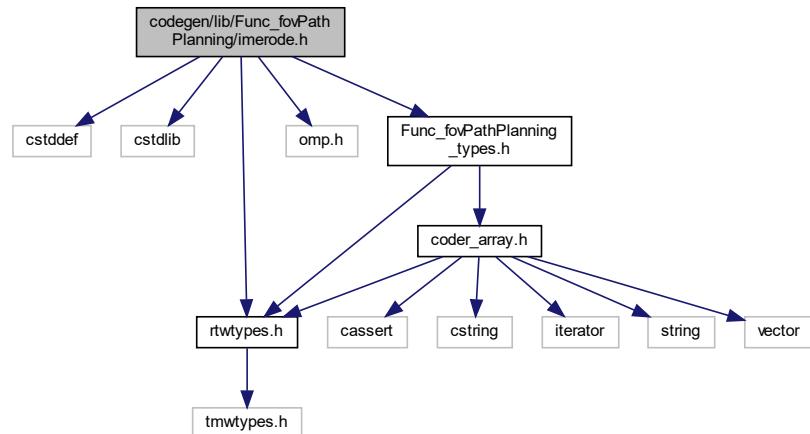
224 static unsigned long long Apadpack[6191444];
225 boolean_T b_nhood[5];
226 for (j = 0; j < 9; j++) {
227   for (i = 0; i < 3029; i++) {
228     Apadpre[i + 3029 * j] = MAX_uint64_T;
229   }
230 }
231
232 for (j = 0; j < 2020; j++) {
233   for (i = 0; i < 9; i++) {
234     Apadpre[i + 3029 * (j + 9)] = MAX_uint64_T;
235   }
236
237 std::memcpy(&Apadpre[j * 3029 + 27270], &A[j * 3020], 3020U * sizeof
238           (unsigned long long));
239 }
240
241 for (j = 0; j < 9; j++) {
242   for (i = 0; i < 3038; i++) {
243     Apad[i + 3038 * (j + 2029)] = MAX_uint64_T;
244   }
245 }
246
247 for (j = 0; j < 2029; j++) {
248   for (i = 0; i < 9; i++) {
249     Apad[(i + 3038 * j) + 3029] = MAX_uint64_T;
250   }
251
252 std::memcpy(&Apad[j * 3038], &Apadpre[j * 3029], 3029U * sizeof(unsigned
253           long long));
254 }
255
256 for (i = 0; i < 7; i++) {
257   nhood[i] = true;
258 }
259
260 asizeT[0] = 3038.0;
261 nsizeT[0] = 7.0;
262 asizeT[1] = 2038.0;
263 nsizeT[1] = 1.0;
264 erode_fflat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
265 asizeT[0] = 3038.0;
266 nsizeT[0] = 5.0;
267 asizeT[1] = 2038.0;
268 nsizeT[1] = 5.0;
269 erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
270 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
271 for (i = 0; i < 7; i++) {
272   nhood[i] = true;
273 }
274
275 asizeT[0] = 3038.0;
276 nsizeT[0] = 1.0;
277 asizeT[1] = 2038.0;
278 nsizeT[1] = 7.0;
279 erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
280 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
281 asizeT[0] = 3038.0;
282 nsizeT[0] = 5.0;
283 asizeT[1] = 2038.0;
284 nsizeT[1] = 5.0;
285 erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bv1, nsizeT, 2.0, Apad);
286 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
287 for (j = 0; j < 5; j++) {
288   b_nhood[j] = true;
289 }
290
291 asizeT[0] = 3038.0;
292 nsizeT[0] = 1.0;
293 asizeT[1] = 2038.0;
294 nsizeT[1] = 5.0;
295 erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
296 std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
297 for (j = 0; j < 5; j++) {
298   b_nhood[j] = true;
299 }
300
301 asizeT[0] = 3038.0;
302 nsizeT[0] = 5.0;
303 asizeT[1] = 2038.0;
304 nsizeT[1] = 1.0;
305 erode_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
306 for (j = 0; j < 2020; j++) {
307   std::memcpy(&B[j * 3020], &Apad[j * 3038 + 27351], 3020U * sizeof(unsigned
308           long long));
309 }
310 }
```

Here is the caller graph for this function:

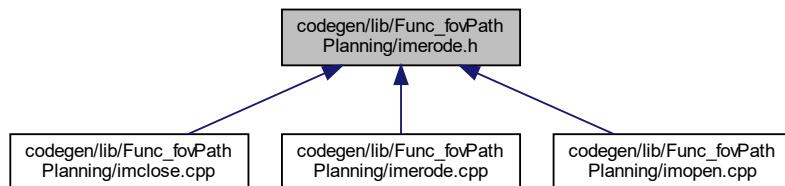


7.26 codegen/lib/Func_fovPathPlanning/imerode.h File Reference

```
#include <cstdint>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for imerode.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void b_imerode (const unsigned long long A[6000000], unsigned long long B[6000000])`
- `void c_imerode (const unsigned long long A[6201600], unsigned long long B[6201600])`
- `void imerode (const unsigned long long A[6100400], unsigned long long B[6100400])`

7.26.1 Macro Definition Documentation

7.26.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file imerode.h.

7.26.2 Function Documentation

7.26.2.1 b_imerode()

```
void b_imerode (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

Definition at line 21 of file imerode.cpp.

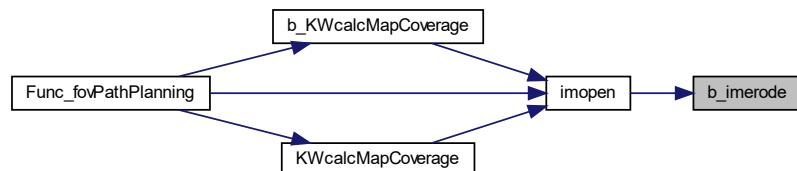
```
22 {
23     int j;
24     int i;
25     static unsigned long long Apadpre[6045081];
26     static unsigned long long Apad[6090324];
27     double asizeT[2];
28     boolean_T nhood[7];
29     double nsizeT[2];
30     static unsigned long long Apadpack[6090324];
31     boolean_T b_nhood[5];
32     for (j = 0; j < 9; j++) {
33         for (i = 0; i < 3009; i++) {
34             Apadpre[i + 3009 * j] = MAX_uint64_T;
35         }
36     }
37     for (j = 0; j < 2000; j++) {
38         for (i = 0; i < 9; i++) {
39             Apadpre[i + 3009 * (j + 9)] = MAX_uint64_T;
40         }
41     }
42     std::memcpy(&Apadpre[j * 3009 + 27090], &A[j * 3000], 3000U * sizeof
43                 (unsigned long long));
44 }
45
46
47     for (j = 0; j < 9; j++) {
48         for (i = 0; i < 3018; i++) {
49             Apad[i + 3018 * (j + 2009)] = MAX_uint64_T;
```

```

50     }
51 }
52
53 for (j = 0; j < 2009; j++) {
54     for (i = 0; i < 9; i++) {
55         Apad[(i + 3018 * j) + 3009] = MAX_uint64_T;
56     }
57
58     std::memcpy(&Apad[j * 3018], &Apadpre[j * 3009], 3009U * sizeof(unsigned
59     long long));
60 }
61
62 for (i = 0; i < 7; i++) {
63     nhood[i] = true;
64 }
65
66 asizeT[0] = 3018.0;
67 nsizet[0] = 7.0;
68 asizeT[1] = 2018.0;
69 nsizet[1] = 1.0;
70 erode_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizet, 2.0, Apadpack);
71 asizeT[0] = 3018.0;
72 nsizet[0] = 5.0;
73 asizeT[1] = 2018.0;
74 nsizet[1] = 5.0;
75 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizet, 2.0, Apad);
76 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
77 for (i = 0; i < 7; i++) {
78     nhood[i] = true;
79 }
80
81 asizeT[0] = 3018.0;
82 nsizet[0] = 1.0;
83 asizeT[1] = 2018.0;
84 nsizet[1] = 7.0;
85 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizet, 2.0, Apad);
86 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
87 asizeT[0] = 3018.0;
88 nsizet[0] = 5.0;
89 asizeT[1] = 2018.0;
90 nsizet[1] = 5.0;
91 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv1, nsizet, 2.0, Apad);
92 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
93 for (j = 0; j < 5; j++) {
94     b_nhood[j] = true;
95 }
96
97 asizeT[0] = 3018.0;
98 nsizet[0] = 1.0;
99 asizeT[1] = 2018.0;
100 nsizet[1] = 5.0;
101 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizet, 2.0, Apad);
102 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
103 for (j = 0; j < 5; j++) {
104     b_nhood[j] = true;
105 }
106
107 asizeT[0] = 3018.0;
108 nsizet[0] = 5.0;
109 asizeT[1] = 2018.0;
110 nsizet[1] = 1.0;
111 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizet, 2.0, Apad);
112 for (j = 0; j < 2000; j++) {
113     std::memcpy(&B[j * 3000], &Apad[j * 3018 + 27171], 3000U * sizeof(unsigned
114     long long));
115 }
116 }

```

Here is the caller graph for this function:



7.26.2.2 c_imerode()

```
void c_imerode (
    const unsigned long long A[6201600],
    unsigned long long B[6201600] )
```

Definition at line 118 of file imerode.cpp.

```
119 {
120     int j;
121     int i;
122     static unsigned long long Apadpre[6298481];
123     static unsigned long long Apad[6396084];
124     double asizeT[2];
125     boolean_T nhood[15];
126     double nsizeT[2];
127     static unsigned long long Apadpack[6396084];
128     boolean_T b_nhood[5];
129     for (j = 0; j < 19; j++) {
130         for (i = 0; i < 3059; i++) {
131             Apadpre[i + 3059 * j] = MAX_uint64_T;
132         }
133     }
134
135     for (j = 0; j < 2040; j++) {
136         for (i = 0; i < 19; i++) {
137             Apadpre[i + 3059 * (j + 19)] = MAX_uint64_T;
138         }
139
140         std::memcpy(&Apadpre[j * 3059 + 58140], &A[j * 3040], 3040U * sizeof
141                         (unsigned long long));
142     }
143
144     for (j = 0; j < 19; j++) {
145         for (i = 0; i < 3078; i++) {
146             Apad[i + 3078 * (j + 2059)] = MAX_uint64_T;
147         }
148     }
149
150     for (j = 0; j < 2059; j++) {
151         for (i = 0; i < 19; i++) {
152             Apad[(i + 3078 * j) + 3059] = MAX_uint64_T;
153         }
154
155         std::memcpy(&Apad[j * 3078], &Apadpre[j * 3059], 3059U * sizeof(unsigned
156                         long long));
157     }
158
159     for (i = 0; i < 15; i++) {
160         nhood[i] = true;
161     }
162
163     asizeT[0] = 3078.0;
164     nsizeT[0] = 15.0;
165     asizeT[1] = 2078.0;
166     nsizeT[1] = 1.0;
167     erode_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
168     asizeT[0] = 3078.0;
169     nsizeT[0] = 11.0;
170     asizeT[1] = 2078.0;
171     nsizeT[1] = 11.0;
172     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv2, nsizeT, 2.0, Apad);
173     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
174     for (i = 0; i < 15; i++) {
175         nhood[i] = true;
176     }
177
178     asizeT[0] = 3078.0;
179     nsizeT[0] = 1.0;
180     asizeT[1] = 2078.0;
181     nsizeT[1] = 15.0;
182     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
183     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
184     asizeT[0] = 3078.0;
185     nsizeT[0] = 11.0;
186     asizeT[1] = 2078.0;
187     nsizeT[1] = 11.0;
188     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv3, nsizeT, 2.0, Apad);
189     std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
190     for (j = 0; j < 5; j++) {
```

```

191     b_nhood[j] = true;
192 }
193
194 asizeT[0] = 3078.0;
195 nsizeT[0] = 1.0;
196 asizeT[1] = 2078.0;
197 nsizeT[1] = 5.0;
198 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
199 std::memcpy(&Apadpack[0], &Apad[0], 6396084U * sizeof(unsigned long long));
200 for (j = 0; j < 5; j++) {
201     b_nhood[j] = true;
202 }
203
204 asizeT[0] = 3078.0;
205 nsizeT[0] = 5.0;
206 asizeT[1] = 2078.0;
207 nsizeT[1] = 1.0;
208 erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
209 for (j = 0; j < 2040; j++) {
210     std::memcpy(&B[j * 3040], &Apad[j * 3078 + 58501], 3040U * sizeof(unsigned
211         long long));
212 }
213 }
```

Here is the caller graph for this function:



7.26.2.3 imerode()

```

void imerode (
    const unsigned long long A[6100400],
    unsigned long long B[6100400] )
```

Definition at line 215 of file imerode.cpp.

```

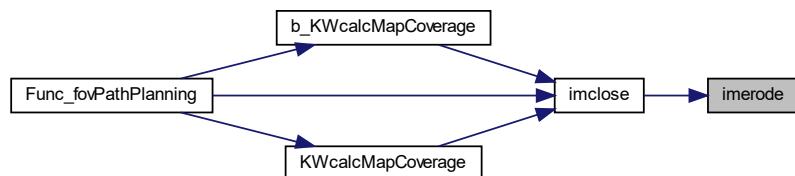
216 {
217     int j;
218     int i;
219     static unsigned long long Apadpre[6145841];
220     static unsigned long long Apad[6191444];
221     double asizeT[2];
222     boolean_T nhood[7];
223     double nsizeT[2];
224     static unsigned long long Apadpack[6191444];
225     boolean_T b_nhood[5];
226     for (j = 0; j < 9; j++) {
227         for (i = 0; i < 3029; i++) {
228             Apadpre[i + 3029 * j] = MAX_uint64_T;
229         }
230     }
231
232     for (j = 0; j < 2020; j++) {
233         for (i = 0; i < 9; i++) {
234             Apadpre[i + 3029 * (j + 9)] = MAX_uint64_T;
235         }
236
237         std::memcpy(&Apadpre[j * 3029 + 27270], &A[j * 3020], 3020U * sizeof(
238             unsigned long long));
239     }
240
241     for (j = 0; j < 9; j++) {
242         for (i = 0; i < 3038; i++) {
243             Apad[i + 3038 * (j + 2029)] = MAX_uint64_T;
244         }
245     }
```

```

246
247     for (j = 0; j < 2029; j++) {
248         for (i = 0; i < 9; i++) {
249             Apad[(i + 3038 * j) + 3029] = MAX_uint64_T;
250         }
251
252         std::memcpy(&Apad[j * 3038], &Apadpre[j * 3029], 3029U * sizeof(unsigned
253                                     long long));
254     }
255
256     for (i = 0; i < 7; i++) {
257         nhood[i] = true;
258     }
259
260     asizeT[0] = 3038.0;
261     nsizeT[0] = 7.0;
262     asizeT[1] = 2038.0;
263     nsizeT[1] = 1.0;
264     erode_flat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
265     asizeT[0] = 3038.0;
266     nsizeT[0] = 5.0;
267     asizeT[1] = 2038.0;
268     nsizeT[1] = 5.0;
269     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
270     std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
271     for (i = 0; i < 7; i++) {
272         nhood[i] = true;
273     }
274
275     asizeT[0] = 3038.0;
276     nsizeT[0] = 1.0;
277     asizeT[1] = 2038.0;
278     nsizeT[1] = 7.0;
279     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
280     std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
281     asizeT[0] = 3038.0;
282     nsizeT[0] = 5.0;
283     asizeT[1] = 2038.0;
284     nsizeT[1] = 5.0;
285     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, bvl, nsizeT, 2.0, Apad);
286     std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
287     for (j = 0; j < 5; j++) {
288         b_nhood[j] = true;
289     }
290
291     asizeT[0] = 3038.0;
292     nsizeT[0] = 1.0;
293     asizeT[1] = 2038.0;
294     nsizeT[1] = 5.0;
295     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
296     std::memcpy(&Apadpack[0], &Apad[0], 6191444U * sizeof(unsigned long long));
297     for (j = 0; j < 5; j++) {
298         b_nhood[j] = true;
299     }
300
301     asizeT[0] = 3038.0;
302     nsizeT[0] = 5.0;
303     asizeT[1] = 2038.0;
304     nsizeT[1] = 1.0;
305     erode_flat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
306     for (j = 0; j < 2020; j++) {
307         std::memcpy(&B[j * 3020], &Apad[j * 3038 + 27351], 3020U * sizeof(unsigned
308                                     long long));
309     }
310 }

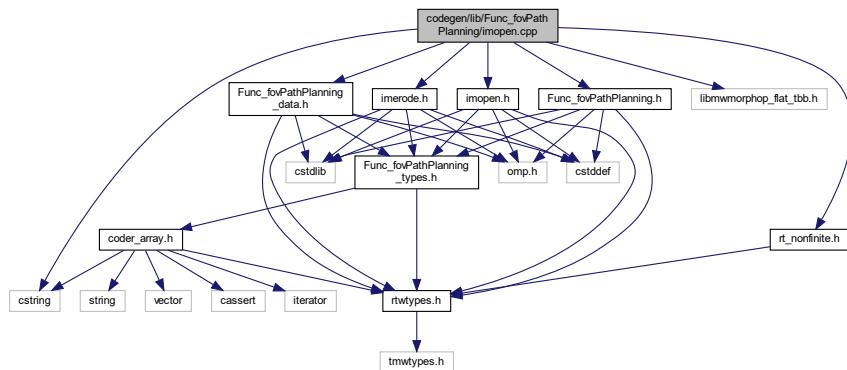
```

Here is the caller graph for this function:



7.27 codegen/lib/Func_fovPathPlanning/imopen.cpp File Reference

```
#include "imopen.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_data.h"
#include "imerode.h"
#include "libmwmorphop_flat_tbb.h"
#include "rt_nonfinite.h"
#include <cstring>
Include dependency graph for imopen.cpp:
```



Functions

- void `imopen` (const unsigned long long A[6000000], unsigned long long B[6000000])

7.27.1 Function Documentation

7.27.1.1 `imopen()`

```
void imopen (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

Definition at line 22 of file `imopen.cpp`.

```
23 {
24     static unsigned long long b_A[6000000];
25     int i;
26     static unsigned long long Apadpre[6045081];
27     static unsigned long long Apad[6090324];
28     double asizeT[2];
29     boolean_T nhood[7];
30     double nsizeT[2];
31     static unsigned long long Apadpack[6090324];
32     boolean_T b_nhhood[5];
33     b_imero(A, b_A);
34     for (i = 0; i < 9; i++) {
35         std::memset(&Apadpre[i * 3009], 0, 3009U * sizeof(unsigned long long));
36     }
37     for (i = 0; i < 2000; i++) {
```

```

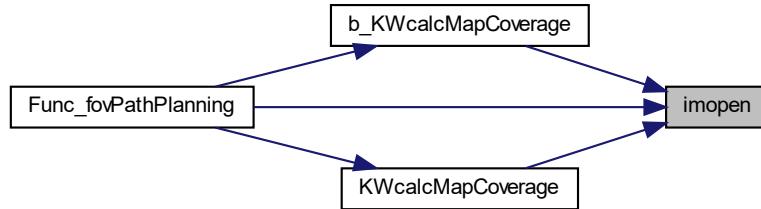
39     std::memset(&Apadpre[i * 3009 + 27081], 0, 9U * sizeof(unsigned long long));
40     std::memcpy(&Apadpre[i * 3009 + 27090], &b_A[i * 3000], 3000U * sizeof
41             (unsigned long long));
42 }
43
44 for (i = 0; i < 9; i++) {
45     std::memset(&Apad[i * 3018 + 6063162], 0, 3018U * sizeof(unsigned long long));
46 }
47
48 for (i = 0; i < 2009; i++) {
49     std::memset(&Apad[i * 3018 + 3009], 0, 9U * sizeof(unsigned long long));
50     std::memcpy(&Apad[i * 3018], &Apadpre[i * 3009], 3009U * sizeof(unsigned
51             long long));
52 }
53
54 for (i = 0; i < 7; i++) {
55     nhood[i] = true;
56 }
57
58 asizeT[0] = 3018.0;
59 nsizeT[0] = 7.0;
60 asizeT[1] = 2018.0;
61 nsizeT[1] = 1.0;
62 dilate_fflat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
63 asizeT[0] = 3018.0;
64 nsizeT[0] = 5.0;
65 asizeT[1] = 2018.0;
66 nsizeT[1] = 5.0;
67 dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
68 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
69 for (i = 0; i < 7; i++) {
70     nhood[i] = true;
71 }
72
73 asizeT[0] = 3018.0;
74 nsizeT[0] = 1.0;
75 asizeT[1] = 2018.0;
76 nsizeT[1] = 7.0;
77 dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
78 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
79 asizeT[0] = 3018.0;
80 nsizeT[0] = 5.0;
81 asizeT[1] = 2018.0;
82 nsizeT[1] = 5.0;
83 dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bvl, nsizeT, 2.0, Apad);
84 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
85 for (i = 0; i < 5; i++) {
86     b_nhood[i] = true;
87 }
88
89 asizeT[0] = 3018.0;
90 nsizeT[0] = 1.0;
91 asizeT[1] = 2018.0;
92 nsizeT[1] = 5.0;
93 dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
94 std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
95 for (i = 0; i < 5; i++) {
96     b_nhood[i] = true;
97 }
98
99 asizeT[0] = 3018.0;
100 nsizeT[0] = 5.0;
101 asizeT[1] = 2018.0;
102 nsizeT[1] = 1.0;
103 dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
104 for (i = 0; i < 2000; i++) {
105     std::memcpy(&B[i * 3000], &Apad[i * 3018 + 27171], 3000U * sizeof(unsigned
106             long long));
107 }
108 }

```

Here is the call graph for this function:

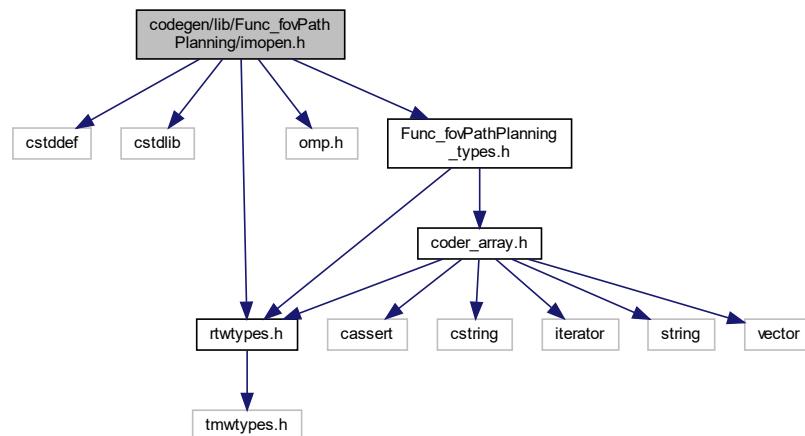


Here is the caller graph for this function:

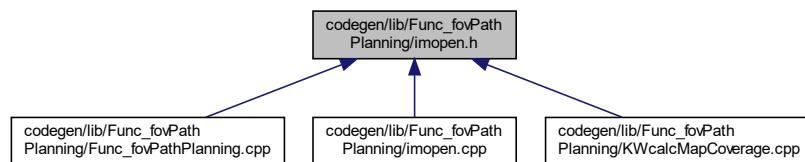


7.28 codegen/lib(Func_fovPathPlanning/imopen.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for imopen.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void imopen (const unsigned long long A[6000000], unsigned long long B[6000000])`

7.28.1 Macro Definition Documentation

7.28.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file imopen.h.

7.28.2 Function Documentation

7.28.2.1 imopen()

```
void imopen (
    const unsigned long long A[6000000],
    unsigned long long B[6000000] )
```

Definition at line 22 of file imopen.cpp.

```
23 {
24     static unsigned long long b_A[6000000];
25     int i;
26     static unsigned long long Apadpre[6045081];
27     static unsigned long long Apad[6090324];
28     double asizeT[2];
29     boolean_T nhood[7];
30     double nsizeT[2];
31     static unsigned long long Apadpack[6090324];
32     boolean_T b_nhood[5];
33     b_imerode(A, b_A);
34     for (i = 0; i < 9; i++) {
35         std::memset(&Apadpre[i * 3009], 0, 3009U * sizeof(unsigned long long));
36     }
37
38     for (i = 0; i < 2000; i++) {
39         std::memset(&Apadpre[i * 3009 + 27081], 0, 9U * sizeof(unsigned long long));
40         std::memcpy(&Apadpre[i * 3009 + 27090], &b_A[i * 3000], 3000U * sizeof
41                     (unsigned long long));
42     }
43
44     for (i = 0; i < 9; i++) {
45         std::memset(&Apad[i * 3018 + 6063162], 0, 3018U * sizeof(unsigned long long));
46     }
47
48     for (i = 0; i < 2009; i++) {
49         std::memset(&Apad[i * 3018 + 3009], 0, 9U * sizeof(unsigned long long));
50         std::memcpy(&Apad[i * 3018], &Apadpre[i * 3009], 3009U * sizeof(unsigned
51                     long long));
52     }
53 }
```

```

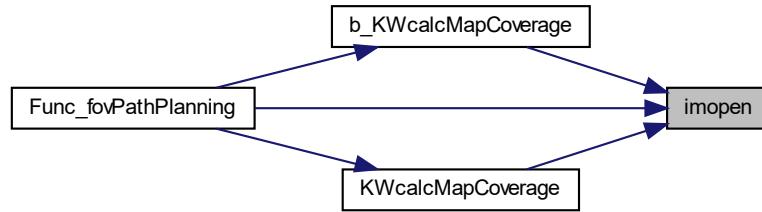
54     for (i = 0; i < 7; i++) {
55         nhood[i] = true;
56     }
57
58     asizeT[0] = 3018.0;
59     nsizeT[0] = 7.0;
60     asizeT[1] = 2018.0;
61     nsizeT[1] = 1.0;
62     dilate_fflat_uint64_tbb(Apad, asizeT, 2.0, nhood, nsizeT, 2.0, Apadpack);
63     asizeT[0] = 3018.0;
64     nsizeT[0] = 5.0;
65     asizeT[1] = 2018.0;
66     nsizeT[1] = 5.0;
67     dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bv, nsizeT, 2.0, Apad);
68     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
69     for (i = 0; i < 7; i++) {
70         nhood[i] = true;
71     }
72
73     asizeT[0] = 3018.0;
74     nsizeT[0] = 1.0;
75     asizeT[1] = 2018.0;
76     nsizeT[1] = 7.0;
77     dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, nhood, nsizeT, 2.0, Apad);
78     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
79     asizeT[0] = 3018.0;
80     nsizeT[0] = 5.0;
81     asizeT[1] = 2018.0;
82     nsizeT[1] = 5.0;
83     dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, bvl, nsizeT, 2.0, Apad);
84     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
85     for (i = 0; i < 5; i++) {
86         b_nhood[i] = true;
87     }
88
89     asizeT[0] = 3018.0;
90     nsizeT[0] = 1.0;
91     asizeT[1] = 2018.0;
92     nsizeT[1] = 5.0;
93     dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
94     std::memcpy(&Apadpack[0], &Apad[0], 6090324U * sizeof(unsigned long long));
95     for (i = 0; i < 5; i++) {
96         b_nhood[i] = true;
97     }
98
99     asizeT[0] = 3018.0;
100    nsizeT[0] = 5.0;
101    asizeT[1] = 2018.0;
102    nsizeT[1] = 1.0;
103    dilate_fflat_uint64_tbb(Apadpack, asizeT, 2.0, b_nhood, nsizeT, 2.0, Apad);
104    for (i = 0; i < 2000; i++) {
105        std::memcpy(&B[i * 3000], &Apad[i * 3018 + 27171], 3000U * sizeof(unsigned
106        long long));
107    }
108 }

```

Here is the call graph for this function:



Here is the caller graph for this function:

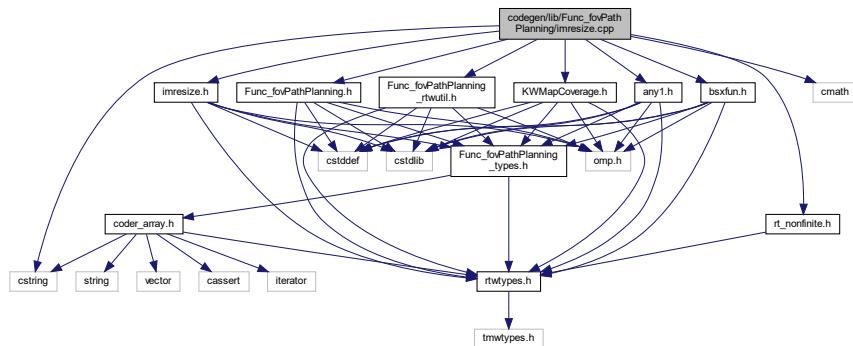


7.29 codegen/lib(Func_fovPathPlanning/imresize.cpp File Reference

```

#include "imresize.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "KWMapCoverage.h"
#include "any1.h"
#include "bsxfun.h"
#include "rt_nonfinite.h"
#include <cmath>
#include <cstring>
  
```

Include dependency graph for imresize.cpp:



Functions

- void [b_imresize](#) (const unsigned long long Ain[6000000], unsigned long long Bout_data[], int Bout_size[2])
- void [imresize](#) (const unsigned long long Ain[6000000], unsigned long long Bout[2542])

7.29.1 Function Documentation

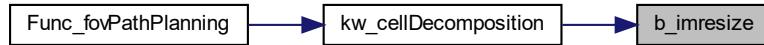
7.29.1.1 b_imresize()

```
void b_imresize (
    const unsigned long long Ain[6000000],
    unsigned long long Bout_data[],
    int Bout_size[2] )
```

Definition at line 609 of file imresize.cpp.

```
611 {
612     double weights_data[186];
613     int weights_size[2];
614     int indices_data[186];
615     int indices_size[2];
616     double b_weights_data[186];
617     int b_weights_size[2];
618     int b_indices_data[186];
619     coder::array<unsigned long long, 2U> r;
620
621     // Resize first dimension
622     contributions(62.0, weights_data, weights_size, indices_data, indices_size);
623
624     // Resize second dimension
625     b_contributions(41.0, b_weights_data, b_weights_size, b_indices_data,
626                      indices_size);
627     c_resizeAlongDim(Ain, weights_data, weights_size, indices_data, r);
628     d_resizeAlongDim(r, b_weights_data, b_weights_size, b_indices_data, Bout_data,
629                      Bout_size);
630 }
```

Here is the caller graph for this function:



7.29.1.2 imresize()

```
void imresize (
    const unsigned long long Ain[6000000],
    unsigned long long Bout[2542] )
```

Definition at line 632 of file imresize.cpp.

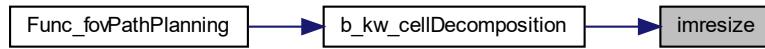
```
634 {
635     int i;
636     int indices[186];
637     static const short iv[186] = { 24, 73, 122, 171, 220, 269, 318, 367, 416, 465,
638         514, 563, 612, 661, 710, 759, 808, 857, 906, 955, 1004, 1053, 1102, 1151,
639         1200, 1249, 1298, 1347, 1396, 1445, 1494, 1543, 1592, 1641, 1690, 1739, 1788,
640         1837, 1886, 1935, 1984, 2033, 2082, 2131, 2180, 2229, 2278, 2327, 2376, 2425,
641         2474, 2523, 2572, 2621, 2670, 2719, 2768, 2817, 2866, 2915, 2964, 3013, 25,
642         74, 123, 172, 221, 270, 319, 368, 417, 466, 515, 564, 613, 662, 711, 760,
643         809, 858, 907, 956, 1005, 1054, 1103, 1152, 1201, 1250, 1299, 1348, 1397,
644         1446, 1495, 1544, 1593, 1642, 1691, 1740, 1789, 1838, 1887, 1936, 1985, 2034,
645         2083, 2132, 2181, 2230, 2279, 2328, 2377, 2426, 2475, 2524, 2573, 2622, 2671,
646         2720, 2769, 2818, 2867, 2916, 2965, 3014, 26, 75, 124, 173, 222, 271, 320,
647         369, 418, 467, 516, 565, 614, 663, 712, 761, 810, 859, 908, 957, 1006, 1055,
648         1104, 1153, 1202, 1251, 1300, 1349, 1398, 1447, 1496, 1545, 1594, 1643, 1692,
649         1741, 1790, 1839, 1888, 1937, 1986, 2035, 2084, 2133, 2182, 2231, 2280, 2329,
650         2378, 2427, 2476, 2525, 2574, 2623, 2672, 2721, 2770, 2819, 2868, 2917, 2966,
651         3015 };
652
653     int aux[6000];
```

```

654     static unsigned long long uv[124000];
655     int r;
656
657     // Resize first dimension
658     // Resize second dimension
659     // Contributions, using pixel indices
660     for (i = 0; i < 186; i++) {
661         indices[i] = iv[i];
662     }
663
664     // Create the auxiliary matrix:
665     aux[0] = 1;
666     aux[3000] = 3000;
667     for (i = 0; i < 2999; i++) {
668         aux[i + 1] = aux[i] + 1;
669         aux[i + 3001] = aux[i + 3000] - 1;
670     }
671
672     // Mirror the out-of-bounds indices using mod:
673     for (i = 0; i < 186; i++) {
674         if (static_cast<double>(indices[i]) - 1.0 == 0.0) {
675             r = 0;
676         } else {
677             r = static_cast<int>(std::fmod(static_cast<double>(indices[i]) - 1.0,
678                                         6000.0));
679             if ((r != 0) && (static_cast<double>(indices[i]) - 1.0 < 0.0)) {
680                 r += 6000;
681             }
682         }
683         indices[i] = aux[r];
684     }
685
686     resizeAlongDim(Ain, *(int (*)[62])&indices[62], uv);
687     b_resizeAlongDim(uv, Bout);
688 }

```

Here is the caller graph for this function:



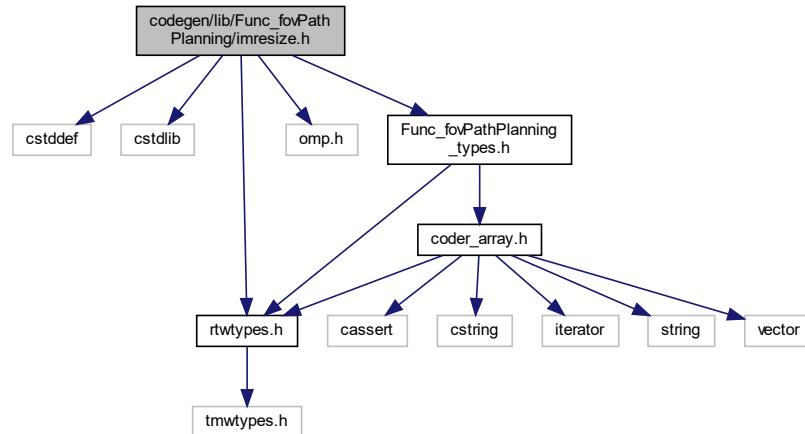
7.30 codegen/lib/Func_fovPathPlanning/imresize.h File Reference

```

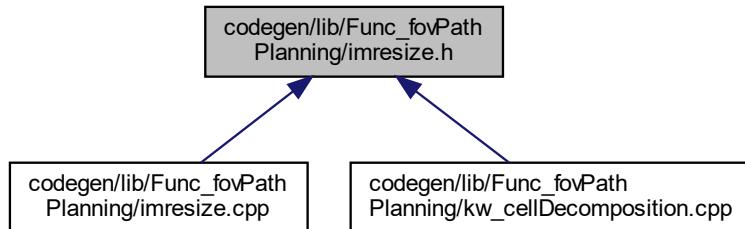
#include <cstdint>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"

```

Include dependency graph for imresize.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void b_imresize (const unsigned long long Ain[6000000], unsigned long long Bout_data[], int Bout_size[2])`
- `void imresize (const unsigned long long Ain[6000000], unsigned long long Bout[2542])`

7.30.1 Macro Definition Documentation

7.30.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file imresize.h.

7.30.2 Function Documentation

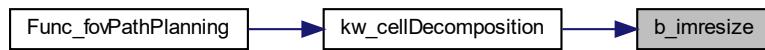
7.30.2.1 b_imresize()

```
void b_imresize (
    const unsigned long long Ain[6000000],
    unsigned long long Bout_data[],
    int Bout_size[2] )
```

Definition at line 609 of file imresize.cpp.

```
611 {
612     double weights_data[186];
613     int weights_size[2];
614     int indices_data[186];
615     int indices_size[2];
616     double b_weights_data[186];
617     int b_weights_size[2];
618     int b_indices_data[186];
619     coder::array<unsigned long long, 2U> r;
620
621     // Resize first dimension
622     contributions(62.0, weights_data, weights_size, indices_data, indices_size);
623
624     // Resize second dimension
625     b_contributions(41.0, b_weights_data, b_weights_size, b_indices_data,
626                     indices_size);
627     c_resizeAlongDim(Ain, weights_data, weights_size, indices_data, r);
628     d_resizeAlongDim(r, b_weights_data, b_weights_size, b_indices_data, Bout_data,
629                      Bout_size);
630 }
```

Here is the caller graph for this function:



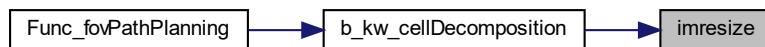
7.30.2.2 imresize()

```
void imresize (
    const unsigned long long Ain[6000000],
    unsigned long long Bout[2542] )
```

Definition at line 632 of file imresize.cpp.

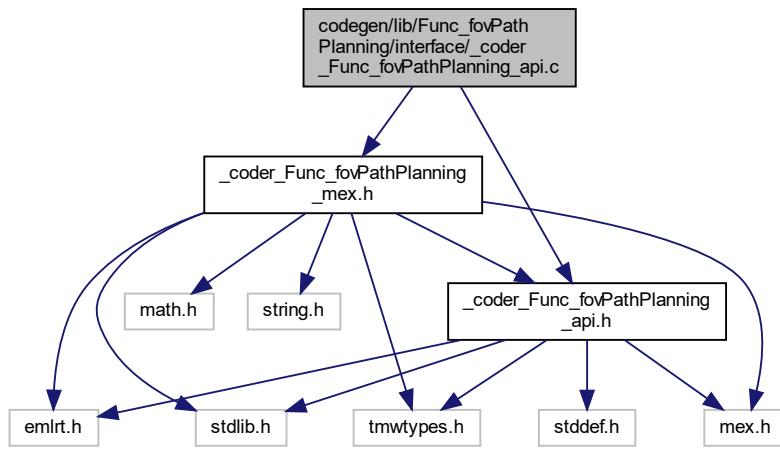
```
634 {
635     int i;
636     int indices[186];
637     static const short iv[186] = { 24, 73, 122, 171, 220, 269, 318, 367, 416, 465,
638         514, 563, 612, 661, 710, 759, 808, 857, 906, 955, 1004, 1053, 1102, 1151,
639         1200, 1249, 1298, 1347, 1396, 1445, 1494, 1543, 1592, 1641, 1690, 1739, 1788,
640         1837, 1886, 1935, 1984, 2033, 2082, 2131, 2180, 2229, 2278, 2327, 2376, 2425,
641         2474, 2523, 2572, 2621, 2670, 2719, 2768, 2817, 2866, 2915, 2964, 3013, 25,
642         74, 123, 172, 221, 270, 319, 368, 417, 466, 515, 564, 613, 662, 711, 760,
643         809, 858, 907, 956, 1005, 1054, 1103, 1152, 1201, 1250, 1299, 1348, 1397,
644         1446, 1495, 1544, 1593, 1642, 1691, 1740, 1789, 1838, 1887, 1936, 1985, 2034,
645         2083, 2132, 2181, 2230, 2279, 2328, 2377, 2426, 2475, 2524, 2573, 2622, 2671,
646         2720, 2769, 2818, 2867, 2916, 2965, 3014, 26, 75, 124, 173, 222, 271, 320,
647         369, 418, 467, 516, 565, 614, 663, 712, 761, 810, 859, 908, 957, 1006, 1055,
648         1104, 1153, 1202, 1251, 1300, 1349, 1398, 1447, 1496, 1545, 1594, 1643, 1692,
649         1741, 1790, 1839, 1888, 1937, 1986, 2035, 2084, 2133, 2182, 2231, 2280, 2329,
650         2378, 2427, 2476, 2525, 2574, 2623, 2672, 2721, 2770, 2819, 2868, 2917, 2966,
651         3015 };
652
653     int aux[6000];
654     static unsigned long long uv[124000];
655     int r;
656
657     // Resize first dimension
658     // Resize second dimension
659     // Contributions, using pixel indices
660     for (i = 0; i < 186; i++) {
661         indices[i] = iv[i];
662     }
663
664     // Create the auxiliary matrix:
665     aux[0] = 1;
666     aux[3000] = 3000;
667     for (i = 0; i < 2999; i++) {
668         aux[i + 1] = aux[i] + 1;
669         aux[i + 3001] = aux[i + 3000] - 1;
670     }
671
672     // Mirror the out-of-bounds indices using mod:
673     for (i = 0; i < 186; i++) {
674         if (static_cast<double>(indices[i]) - 1.0 == 0.0) {
675             r = 0;
676         } else {
677             r = static_cast<int>(std::fmod(static_cast<double>(indices[i]) - 1.0,
678                 6000.0));
679             if ((r != 0) && (static_cast<double>(indices[i]) - 1.0 < 0.0)) {
680                 r += 6000;
681             }
682         }
683         indices[i] = aux[r];
684     }
685
686     resizeAlongDim(Ain, *(int (*)[62])&indices[62], uv);
687     b_resizeAlongDim(uv, Bout);
688 }
689 }
```

Here is the caller graph for this function:



7.31 codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.c File Reference

```
#include "_coder_Func_fovPathPlanning_api.h"
#include "_coder_Func_fovPathPlanning_mex.h"
Include dependency graph for _coder_Func_fovPathPlanning_api.c:
```



Functions

- void `Func_fovPathPlanning_api` (const mxArray *const prhs[4], int32_T nlhs, const mxArray *plhs[2])
- void `Func_fovPathPlanning_atexit` (void)
- void `Func_fovPathPlanning_initialize` (void)
- void `Func_fovPathPlanning_terminate` (void)

Variables

- emlrtCTX `emlrtRootTLSGlobal` = NULL
- emlrtContext `emlrtContextGlobal`

7.31.1 Function Documentation

7.31.1.1 Func_fovPathPlanning_api()

```
void Func_fovPathPlanning_api (
    const mxArray *const prhs[4],
    int32_T nlhs,
    const mxArray * plhs[2] )
```

Definition at line 216 of file _coder_Func_fovPathPlanning_api.c.

```
218 {
219     int64_T size_img;
220     uint64_T (*binaryMap)[6000000];
221     cell_wrap_0 coordix[3];
222     cell_wrap_0 coordiy[3];
223     emlrtStack st = { NULL,
224         /* site */
225         /* tls */
226         /* prev */
227     };
228     st.tls = emlrtRootTLSGlobal;
229
230     /* Marshall function inputs */
231     size_img = emlrt_marshallIn(&st, emlrtAliasP(prhs[0]), "size_img");
232     binaryMap = c_emlrt_marshallIn(&st, emlrtAlias(prhs[1]), "binaryMap");
233     e_emlrt_marshallIn(&st, emlrtAliasP(prhs[2]), "coordix", coordix);
234     e_emlrt_marshallIn(&st, emlrtAliasP(prhs[3]), "coordiy", coordiy);
235
236     /* Invoke the target function */
237     Func_fovPathPlanning(size_img, *binaryMap, coordix, coordiy);
238
239     /* Marshall function outputs */
240     plhs[0] = emlrt_marshallOut(coordix);
241     if (nlhs > 1) {
242         plhs[1] = emlrt_marshallOut(coordiy);
243     }
244 }
```

Here is the caller graph for this function:



7.31.1.2 Func_fovPathPlanning_atexit()

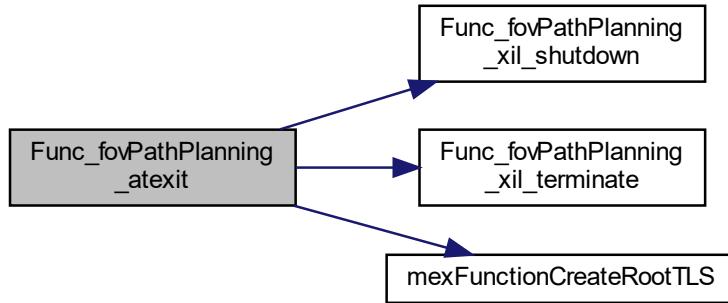
```
void Func_fovPathPlanning_atexit (
    void )
```

Definition at line 246 of file _coder_Func_fovPathPlanning_api.c.

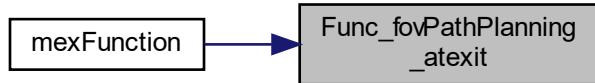
```
247 {
248     emlrtStack st = { NULL,
249         /* site */
250         /* tls */
251         /* prev */
252     };
253     mexFunctionCreateRootTLS();
254     st.tls = emlrtRootTLSGlobal;
255     emlrtEnterRtStackR2012b(&st);
256     emlrtLeaveRtStackR2012b(&st);
257     emlrtDestroyRootTLS(&emlrtRootTLSGlobal);
258     Func_fovPathPlanning_xil_terminate();
259     Func_fovPathPlanning_xil_shutdown();
260     emlrtExitTimeCleanup(&emlrtContextGlobal);
```

```
261 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.31.1.3 Func_fovPathPlanning_initialize()

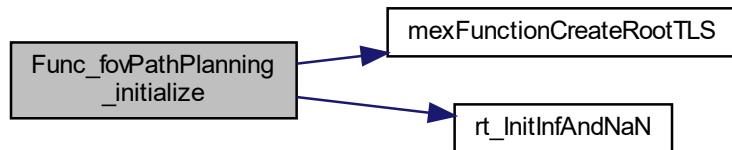
```
void Func_fovPathPlanning_initialize (
    void )
```

Definition at line 263 of file _coder_Func_fovPathPlanning_api.c.

```

264 {
265     emlrtStack st = { NULL,           /* site */
266                     NULL,          /* tls */
267                     NULL,          /* prev */
268     };
269
270     mexFunctionCreateRootTLS();
271     st.tls = emlrtRootTLSGlobal;
272     emlrtClearAllocCountR2012b(&st, false, 0U, 0);
273     emlrtEnterRtStackR2012b(&st);
274     emlrtFirstTimeR2012b(emlrtRootTLSGlobal);
275 }
```

Here is the call graph for this function:



7.31.1.4 Func_fovPathPlanning_terminate()

```
void Func_fovPathPlanning_terminate (
    void )
```

Definition at line 277 of file _coder_Func_fovPathPlanning_api.c.

```
278 {
279     emlrtStack st = { NULL,                         /* site */
280                       NULL,                         /* tls */
281                       NULL,                         /* prev */
282     };
283
284     st.tls = emlrtRootTLSGlobal;
285     emlrtLeaveRtStackR2012b(&st);
286     emlrtDestroyRootTLS(&emlrtRootTLSGlobal);
287 }
```

7.31.2 Variable Documentation

7.31.2.1 emlrtContextGlobal

```
emlrtContext emlrtContextGlobal
```

Initial value:

```
= { true,
  false,
  131594U,
  NULL,
  "Func_fovPathPlanning",
  NULL,
  false,
  { 2045744189U, 2170104910U, 2743257031U, 4284093946U },
  NULL
}
```

Definition at line 18 of file _coder_Func_fovPathPlanning_api.c.

7.31.2.2 emlrtRootTLSGlobal

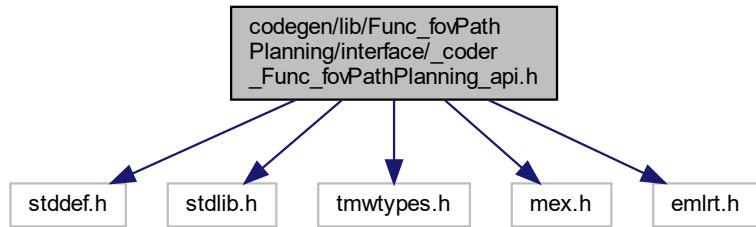
```
emlrtCTX emlrtRootTLSGlobal = NULL
```

Definition at line 17 of file _coder_Func_fovPathPlanning_api.c.

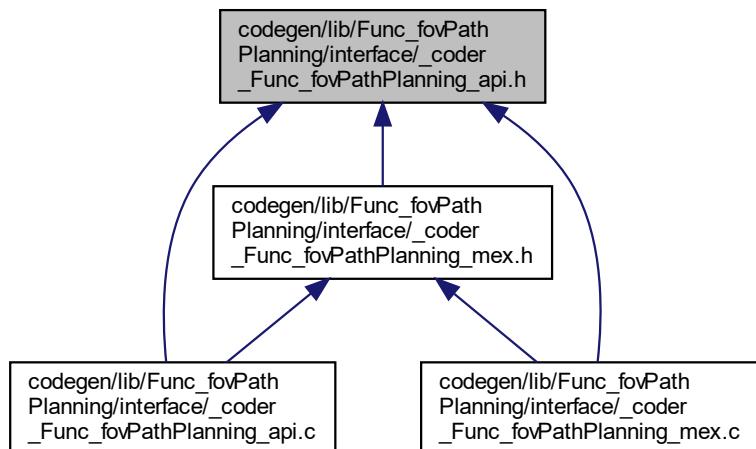
7.32 codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_api.h File Reference

```
#include <stddef.h>
#include <stdlib.h>
#include "tmwtypes.h"
#include "mex.h"
#include "emlrt.h"
```

Include dependency graph for _coder_Func_fovPathPlanning_api.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct `cell_wrap_0`

Macros

- `#define typedef_cell_wrap_0`
- `#define MAX_THREADS omp_get_max_threads()`

Functions

- void `Func_fovPathPlanning` (int64_T size_img, uint64_T binaryMap[6000000], `cell_wrap_0` coordix[3], `cell_wrap_0` coordiy[3])
- void `Func_fovPathPlanning_api` (const mxArray *const prhs[4], int32_T nlhs, const mxArray *plhs[2])
- void `Func_fovPathPlanning_atexit` (void)
- void `Func_fovPathPlanning_initialize` (void)
- void `Func_fovPathPlanning_terminate` (void)
- void `Func_fovPathPlanning_xil_shutdown` (void)
- void `Func_fovPathPlanning_xil_terminate` (void)

Variables

- emlrtCTX `emlrtRootTLSGlobal`
- emlrtContext `emlrtContextGlobal`

7.32.1 Macro Definition Documentation

7.32.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 36 of file _coder_Func_fovPathPlanning_api.h.

7.32.1.2 typedef_cell_wrap_0

```
#define typedef_cell_wrap_0
```

Definition at line 24 of file _coder_Func_fovPathPlanning_api.h.

7.32.2 Function Documentation

7.32.2.1 Func_fovPathPlanning()

```
void Func_fovPathPlanning (
    int64_T size_img,
    uint64_T binaryMap[6000000],
    cell_wrap_0 coordix[3],
    cell_wrap_0 coordiy[3] )
```

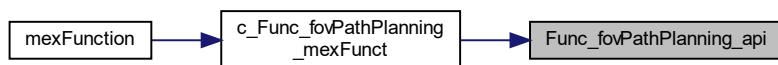
7.32.2.2 Func_fovPathPlanning_api()

```
void Func_fovPathPlanning_api (
    const mxArray *const prhs[4],
    int32_T nlhs,
    const mxArray * plhs[2] )
```

Definition at line 216 of file _coder_Func_fovPathPlanning_api.c.

```
218 {
219     int64_T size_img;
220     uint64_T (*binaryMap)[6000000];
221     cell_wrap_0 coordix[3];
222     cell_wrap_0 coordiy[3];
223     emlrtStack st = { NULL, /* site */
224                       NULL, /* tls */
225                       NULL /* prev */ };
226 };
227
228 st.tls = emlrtRootTLSGlobal;
229
230 /* Marshall function inputs */
231 size_img = emlrt_marshallIn(&st, emlrtAliasP(prhs[0]), "size_img");
232 binaryMap = c_emlrt_marshallIn(&st, emlrtAlias(prhs[1]), "binaryMap");
233 e_emlrt_marshallIn(&st, emlrtAliasP(prhs[2]), "coordix", coordix);
234 e_emlrt_marshallIn(&st, emlrtAliasP(prhs[3]), "coordiy", coordiy);
235
236 /* Invoke the target function */
237 Func_fovPathPlanning(size_img, *binaryMap, coordix, coordiy);
238
239 /* Marshall function outputs */
240 plhs[0] = emlrt_marshallOut(coordix);
241 if (nlhs > 1) {
242     plhs[1] = emlrt_marshallOut(coordiy);
243 }
```

Here is the caller graph for this function:



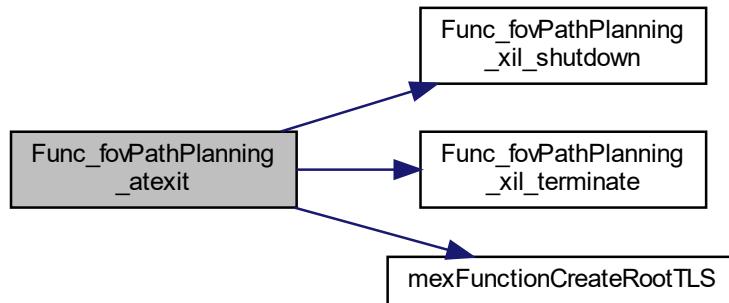
7.32.2.3 Func_fovPathPlanning_atexit()

```
void Func_fovPathPlanning_atexit (
    void )
```

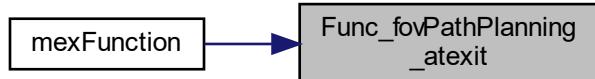
Definition at line 246 of file _coder_Func_fovPathPlanning_api.c.

```
247 {
248     emlrtStack st = { NULL,
249                         /* site */
250                         /* NULL,           */
251                         /* prev */        };
252
253     mexFunctionCreateRootTLS();
254     st.tls = emlrtRootTLSGlobal;
255     emlrtEnterRtStackR2012b(&st);
256     emlrtLeaveRtStackR2012b(&st);
257     emlrtDestroyRootTLS(&emlrtRootTLSGlobal);
258     Func_fovPathPlanning_xil_terminate();
259     Func_fovPathPlanning_xil_shutdown();
260     emlrtExitTimeCleanup(&emlrtContextGlobal);
261 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



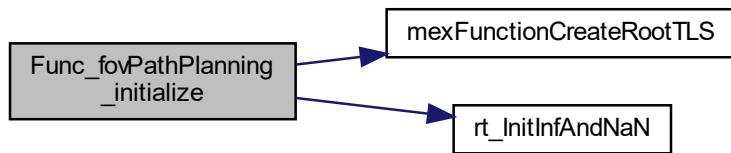
7.32.2.4 Func_fovPathPlanning_initialize()

```
void Func_fovPathPlanning_initialize (
    void )
```

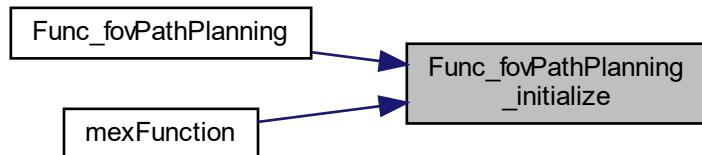
Definition at line 19 of file Func_fovPathPlanning_initialize.cpp.

```
20 {
21     rt_InitInfAndNaN();
22     omp_init_nest_lock(&emlrtNestLockGlobal);
23     isInitialized_Func_fovPathPlanning = true;
24 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



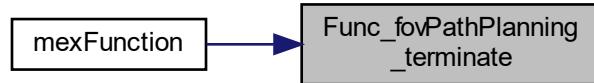
7.32.2.5 Func_fovPathPlanning_terminate()

```
void Func_fovPathPlanning_terminate (
    void )
```

Definition at line 19 of file Func_fovPathPlanning_terminate.cpp.

```
20 {
21     omp_destroy_nest_lock(&emlrtNestLockGlobal);
22     isInitialized_Func_fovPathPlanning = false;
23 }
```

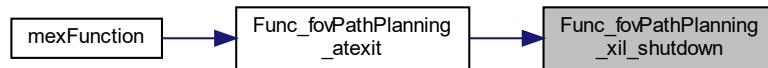
Here is the caller graph for this function:



7.32.2.6 Func_fovPathPlanning_xil_shutdown()

```
void Func_fovPathPlanning_xil_shutdown (
    void )
```

Here is the caller graph for this function:



7.32.2.7 Func_fovPathPlanning_xil_terminate()

```
void Func_fovPathPlanning_xil_terminate (
    void )
```

Here is the caller graph for this function:



7.32.3 Variable Documentation

7.32.3.1 emlrtContextGlobal

```
emlrtContext emlrtContextGlobal [extern]
```

Definition at line 18 of file _coder_Func_fovPathPlanning_api.c.

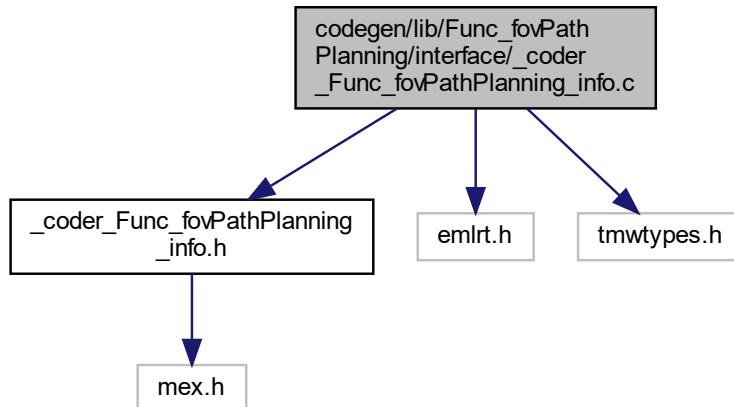
7.32.3.2 emlrtRootTLSGlobal

```
emlrtCTX emlrtRootTLSGlobal [extern]
```

Definition at line 17 of file _coder_Func_fovPathPlanning_api.c.

7.33 codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_info.c File Reference

```
#include "_coder_Func_fovPathPlanning_info.h"
#include "emlrt.h"
#include "tmwtypes.h"
Include dependency graph for _coder_Func_fovPathPlanning_info.c:
```



Functions

- mxArray * [emlrtMexFcnProperties](#) (void)

7.33.1 Function Documentation

7.33.1.1 emlrtMexFcnProperties()

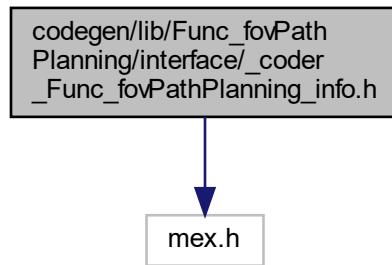
```
mxArray* emlrtMexFcnProperties (
    void )
```

Definition at line 566 of file _coder_Func_fovPathPlanning_info.c.

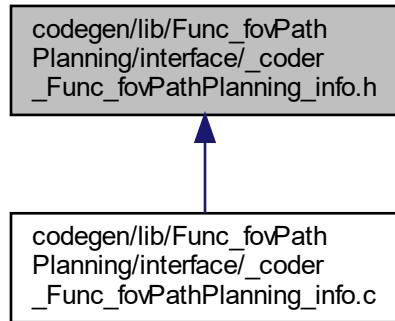
```
567 {
568     mxArray *xResult;
569     mxArray *xEntryPoints;
570     const char * epFieldName[6] = { "Name", "NumberOfInputs", "NumberOfOutputs",
571         "ConstantInputs", "FullPath", "TimeStamp" };
572
573     mxArray *xInputs;
574     const char * propFieldName[4] = { "Version", "ResolvedFunctions",
575         "EntryPoints", "CoverageInfo" };
576
577     xEntryPoints = emlrtCreateStructMatrix(1, 1, 6, epFieldName);
578     xInputs = emlrtCreateLogicalMatrix(1, 4);
579     emlrtSetField(xEntryPoints, 0, "Name", emlrtMxCreateString(
580         "Func_fovPathPlanning"));
581     emlrtSetField(xEntryPoints, 0, "NumberOfInputs", emlrtMxCreateDoubleScalar(4.0));
582     emlrtSetField(xEntryPoints, 0, "NumberOfOutputs", emlrtMxCreateDoubleScalar
583         (2.0));
584     emlrtSetField(xEntryPoints, 0, "ConstantInputs", xInputs);
585     emlrtSetField(xEntryPoints, 0, "FullPath", emlrtMxCreateString(
586         "C:\\\\Users\\\\\\USER\\\\\\Documents\\\\\\\\xc4\xab\\xc4\xab\\xbf\\xc0\\xe5 \\xb9\\xde\\xc0\\xba
587         \\xc6\\xc4\\xc0\\xcf\\\\\\realrealatest (2)\\\\\\realrealatest_c\\\\\\Func_fovPathPlanning.m"));
588     emlrtSetField(xEntryPoints, 0, "TimeStamp", emlrtMxCreateDoubleScalar
589         (738112.83505787037));
590     xResult = emlrtCreateStructMatrix(1, 1, 4, propFieldName);
591     emlrtSetField(xResult, 0, "Version", emlrtMxCreateString(
592         "9.8.0.1451342 (R2020a) Update 5"));
593     emlrtSetField(xResult, 0, "ResolvedFunctions", (mxArray *)
594         emlrtMexFcnResolvedFunctionsInfo());
595     emlrtSetField(xResult, 0, "EntryPoints", xEntryPoints);
596     return xResult;
596 }
```

7.34 codegen/lib/Func_fovPathPlanning/interface/_coder_Func_fov← PathPlanning_info.h File Reference

```
#include "mex.h"
Include dependency graph for _coder_Func_fovPathPlanning_info.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `MEXFUNCTION_LINKAGE mxArray * emlrtMexFcnProperties (void)`

7.34.1 Macro Definition Documentation

7.34.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 17 of file _coder_Func_fovPathPlanning_info.h.

7.34.2 Function Documentation

7.34.2.1 emlrtMexFcnProperties()

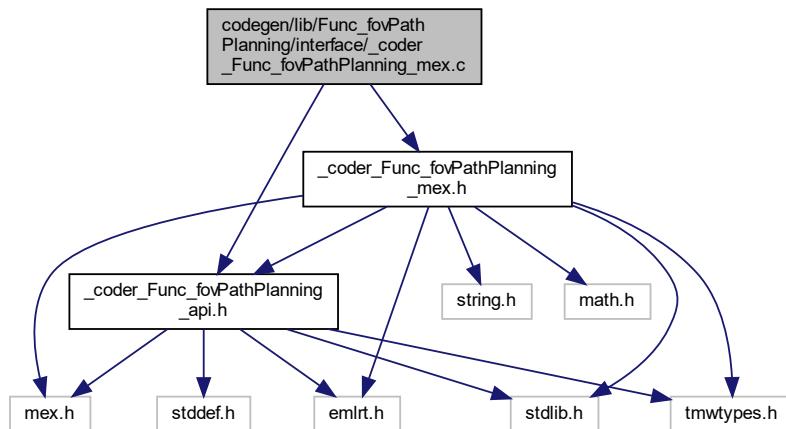
```
MEXFUNCTION_LINKAGE mxArray* emlrtMexFcnProperties (
    void )
```

Definition at line 566 of file _coder_Func_fovPathPlanning_info.c.

```
567 {
568     mxArray *xResult;
569     mxArray *xEntryPoints;
570     const char * epFieldName[6] = { "Name", "NumberOfInputs", "NumberOfOutputs",
571         "ConstantInputs", "FullPath", "TimeStamp" };
572
573     mxArray *xInputs;
574     const char * propFieldName[4] = { "Version", "ResolvedFunctions",
575         "EntryPoints", "CoverageInfo" };
576
577     xEntryPoints = emlrtCreateStructMatrix(1, 1, 6, epFieldName);
578     xInputs = emlrtCreateLogicalMatrix(1, 4);
579     emlrtSetField(xEntryPoints, 0, "Name", emlrtMxCreateString(
580         "Func_fovPathPlanning"));
581     emlrtSetField(xEntryPoints, 0, "NumberOfInputs", emlrtMxCreateDoubleScalar(4.0));
582     emlrtSetField(xEntryPoints, 0, "NumberOfOutputs", emlrtMxCreateDoubleScalar
583         (2.0));
584     emlrtSetField(xEntryPoints, 0, "ConstantInputs", xInputs);
585     emlrtSetField(xEntryPoints, 0, "FullPath", emlrtMxCreateString(
586         "C:\\\\Users\\\\\\USER\\\\\\Documents\\\\\\\\xc4\xab\xab\xbf\xc0\xc5\\xe5 \\xb9\\xde\\xc0\\xba
587         \\xc6\\xc4\\xc0\\xcf\\\\\\realrealatest (2)\\\\\\realrealatest_c\\\\\\Func_fovPathPlanning.m"));
588     emlrtSetField(xEntryPoints, 0, "TimeStamp", emlrtMxCreateDoubleScalar
589         (738112.83505787037));
590     xResult = emlrtCreateStructMatrix(1, 1, 4, propFieldName);
591     emlrtSetField(xResult, 0, "Version", emlrtMxCreateString(
592         "9.8.0.1451342 (R2020a) Update 5"));
593     emlrtSetField(xResult, 0, "ResolvedFunctions", (mxArray *)
594         emlrtMexFcnResolvedFunctionsInfo());
595     emlrtSetField(xResult, 0, "EntryPoints", xEntryPoints);
596 }
```

7.35 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_mex.c File Reference

```
#include "_coder_Func_fovPathPlanning_mex.h"
#include "_coder_Func_fovPathPlanning_api.h"
Include dependency graph for _coder_Func_fovPathPlanning_mex.c:
```



Functions

- MEXFUNCTION_LINKAGE void [c_Func_fovPathPlanning_mexFunct](#) (int32_T nlhs, mxArray *plhs[2], int32_T nrhs, const mxArray *prhs[4])
- void [mexFunction](#) (int32_T nlhs, mxArray *plhs[], int32_T nrhs, const mxArray *prhs[])
- emlrtCTX [mexFunctionCreateRootTLS](#) (void)

7.35.1 Function Documentation

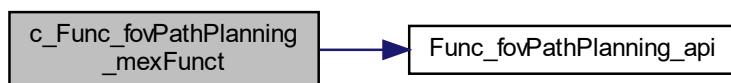
7.35.1.1 c_Func_fovPathPlanning_mexFunct()

```
void c_Func_fovPathPlanning_mexFunct (
    int32_T nlhs,
    mxArray * plhs[2],
    int32_T nrhs,
    const mxArray * prhs[4] )
```

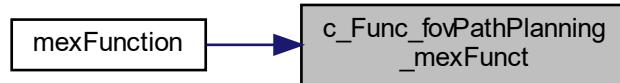
Definition at line 21 of file _coder_Func_fovPathPlanning_mex.c.

```
23 {
24     const mxArray *outputs[2];
25     int32_T b_nlhs;
26     emlrtStack st = { NULL, /* site */
27                       NULL, /* tls */
28                       NULL }; /* prev */
29 };
30
31     st.tls = emlrtRootTLSGlobal;
32
33 /* Check for proper number of arguments. */
34 if (nrhs != 4) {
35     emlrtErrMsgIdAndTxt(&st, "EMLRT:runTime:WrongNumberOfInputs", 5, 12, 4, 4,
36                         20, "Func_fovPathPlanning");
37 }
38
39 if (nlhs > 2) {
40     emlrtErrMsgIdAndTxt(&st, "EMLRT:runTime:TooManyOutputArguments", 3, 4, 20,
41                         "Func_fovPathPlanning");
42 }
43
44 /* Call the function. */
45 Func_fovPathPlanning_api(prhs, nlhs, outputs);
46
47 /* Copy over outputs to the caller. */
48 if (nlhs < 1) {
49     b_nlhs = 1;
50 } else {
51     b_nlhs = nlhs;
52 }
53
54 emlrtReturnArrays(b_nlhs, plhs, outputs);
55 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.35.1.2 mexFunction()

```

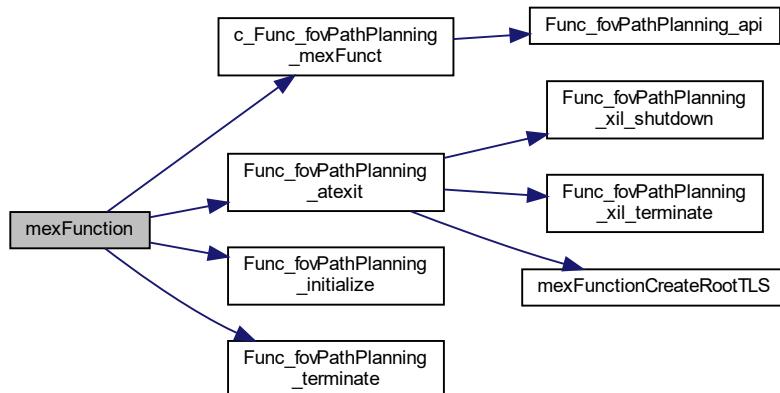
void mexFunction (
    int32_T nlhs,
    mxArray * plhs[],
    int32_T nrhs,
    const mxArray * prhs[] )
  
```

Definition at line 57 of file _coder_Func_fovPathPlanning_mex.c.

```

59 {
60     mexAtExit(&Func_fovPathPlanning_atexit);
61
62     /* Module initialization. */
63     Func_fovPathPlanning_initialize();
64
65     /* Dispatch the entry-point. */
66     c_Func_fovPathPlanning_mexFunct(nlhs, plhs, nrhs, prhs);
67
68     /* Module termination. */
69     Func_fovPathPlanning_terminate();
70 }
  
```

Here is the call graph for this function:



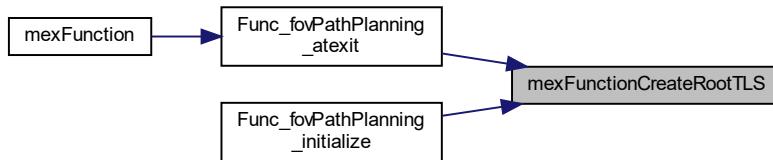
7.35.1.3 mexFunctionCreateRootTLS()

```
emlrtCTX mexFunctionCreateRootTLS (
    void )
```

Definition at line 72 of file _coder_Func_fovPathPlanning_mex.c.

```
73 {
74     emlrtCreateRootTLS(&emlrtRootTLSGlobal, &emlrtContextGlobal, NULL, 1);
75     return emlrtRootTLSGlobal;
76 }
```

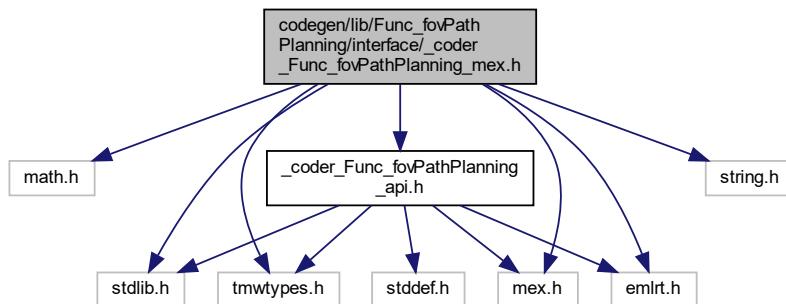
Here is the caller graph for this function:



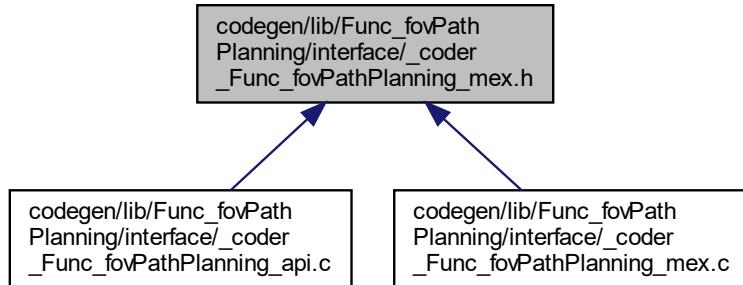
7.36 codegen/lib(Func_fovPathPlanning/interface/_coder_Func_fovPathPlanning_mex.h File Reference

```
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "tmwtypes.h"
#include "mex.h"
#include "emlrt.h"
#include "_coder_Func_fovPathPlanning_api.h"
```

Include dependency graph for _coder_Func_fovPathPlanning_mex.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- MEXFUNCTION_LINKAGE void `mexFunction` (int32_T nlhs, mxArray *plhs[], int32_T nrhs, const mxArray *prhs[])
- emlrtCTX `mexFunctionCreateRootTLS` (void)

7.36.1 Macro Definition Documentation

7.36.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 23 of file `_coder_Func_fovPathPlanning_mex.h`.

7.36.2 Function Documentation

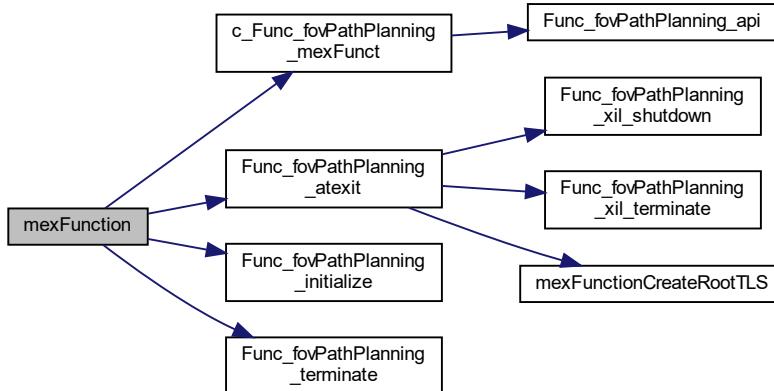
7.36.2.1 mexFunction()

```
MEXFUNCTION_LINKAGE void mexFunction (
    int32_T nlhs,
    mxArray * plhs[],
    int32_T nrhs,
    const mxArray * prhs[] )
```

Definition at line 57 of file _coder_Func_fovPathPlanning_mex.c.

```
59 {
60     mexAtExit(&Func_fovPathPlanning_atexit);
61
62     /* Module initialization. */
63     Func_fovPathPlanning_initialize();
64
65     /* Dispatch the entry-point. */
66     c_Func_fovPathPlanning_mexFunct(nlhs, plhs, nrhs, prhs);
67
68     /* Module termination. */
69     Func_fovPathPlanning_terminate();
70 }
```

Here is the call graph for this function:



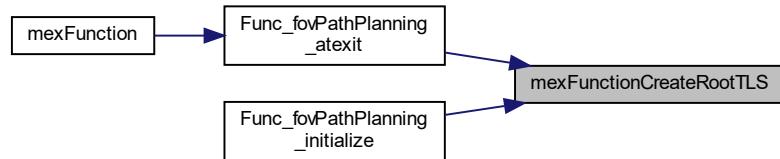
7.36.2.2 mexFunctionCreateRootTLS()

```
emlrtCTX mexFunctionCreateRootTLS (
    void )
```

Definition at line 72 of file _coder_Func_fovPathPlanning_mex.c.

```
73 {
74     emlrtCreateRootTLS(&emlrtRootTLSGlobal, &emlrtContextGlobal, NULL, 1);
75     return emlrtRootTLSGlobal;
76 }
```

Here is the caller graph for this function:

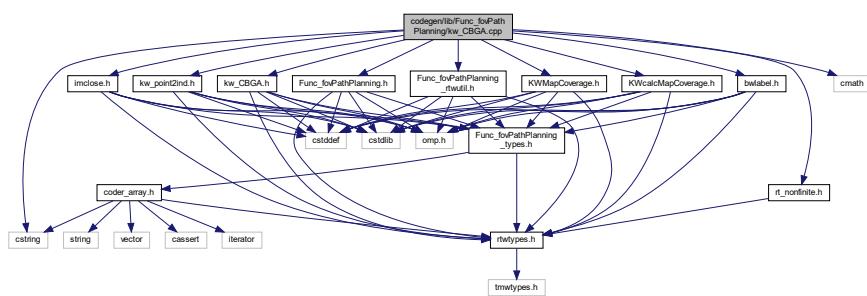


7.37 codegen/lib(Func_fovPathPlanning/kw_CBGA.cpp File Reference

```

#include "kw_CBGA.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "KWMapCoverage.h"
#include "KWCalcMapCoverage.h"
#include "bwlabel.h"
#include "imclose.h"
#include "kw_point2ind.h"
#include "rt_nonfinite.h"
#include <cmath>
#include <cstring>
  
```

Include dependency graph for kw_CBGA.cpp:



Functions

- void **kw_CBGA** (struct_T *sensorParam, unsigned long long coveredMap[6000000], const unsigned long long binaryMap[6000000], double position[2], double th, double Tree[200])

7.37.1 Function Documentation

7.37.1.1 kw_CBGA()

```
void kw_CBGA (
    struct_T * sensorParam,
    unsigned long long coveredMap[6000000],
    const unsigned long long binaryMap[6000000],
    double position[2],
    double th,
    double Tree[200] )
```

Definition at line 26 of file kw_CBGA.cpp.

```
29 {
30     double flag;
31     double cont;
32     double Node[2];
33     static unsigned long long save_map[6000000];
34     int countN;
35     double Cluster_idx_0;
36     double Cluster_idx_1;
37     double Uncovered[2];
38     double new_theta;
39     double num;
40     double sum_y;
41     double distance;
42     coder::array<double, 2U> index2;
43     static unsigned long long label[6000000];
44     static unsigned long long uv[6000000];
45     static double b_label[6000000];
46     int il;
47     static boolean_T x[6000000];
48     coder::array<int, 1U> b_ii;
49     coder::array<short, 1U> jj;
50     coder::array<int, 2U> result;
51     flag = 30.0;
52     cont = 0.0;
53     Node[0] = position[0];
54     Node[1] = position[1];
55     std::memcpy(&save_map[0], &coveredMap[0], 6000000U * sizeof(unsigned long
56     long));
57
58 // position = position + (sensorParam.dvec*3)*[cos(th), sin(th)];
59 std::memset(&Tree[0], 0, 200U * sizeof(double));
60
61 // nearCellcovered t'Á řš
62 countN = -1;
63 Tree[0] = position[0];
64 Cluster_idx_0 = 0.0;
65 Tree[1] = position[1];
66 Cluster_idx_1 = 0.0;
67 int exitgl;
68 do {
69     exitgl = 0;
70     if (flag != 0.0) {
71         double sum_x;
72         double flag_;
73         int i;
74         int idx;
75         int ii;
76         double d;
77         unsigned long long u;
78         cont++;
79         Uncovered[0] = position[0] + 5.0 * std::cos(th);
80         Uncovered[1] = position[1] + 5.0 * std::sin(th);
81         position[0] = Uncovered[0];
82         position[1] = Uncovered[1];
83         new_theta = th;
84
85 // t'ýčyúñ g'g ijd'
86 //     if( frame(position(1), position(2))==0 )
87 //         phase = -phase_;
88 //     return;
89 // end
90 sensorParam->scolor += sensorParam->dc;
91 if (sensorParam->scolor > 150.0) {
92     sensorParam->dc = -1.0;
93 } else {
94     if (sensorParam->scolor < 70.0) {
95         sensorParam->dc = 1.0;
96     }
97 }
98 num = 0.0;
```

```

100     sum_y = 0.0;
101     sum_x = 0.0;
102     flag_ = 0.0;
103     i = static_cast<int>(((th + 0.6108652381980153) + (0.0043633231299858239 -
104     (th - 0.6108652381980153)) / 0.0043633231299858239);
105     idx = 0;
106     int exitg2;
107     do {
108         exitg2 = 0;
109         if (idx <= i - 1) {
110             distance = (th - 0.6108652381980153) + static_cast<double>(idx) *
111                 0.0043633231299858239;
112             Uncovered[0] = position[0] + 300.0 * std::cos(distance);
113             Uncovered[1] = position[1] + 300.0 * std::sin(distance);
114             kw_point2ind(position, Uncovered, index2);
115
116             // Linear indices
117             if (index2.size(1) == 0) {
118                 flag_ = 0.0;
119                 exitg2 = 1;
120             } else {
121                 ii = 0;
122                 while ((ii <= static_cast<int>(index2[0]) - 1) && (binaryMap[ii] >=
123                     200ULL)) {
124                     // flag = flag + (coveredMap(s) > 230);
125                     if (coveredMap[ii] > 230ULL) {
126                         flag_++;
127                         sum_y += std::floor((static_cast<double>(ii) + 1.0) / 3000.0) *
128                             10.0;
129                         sum_x = (sum_x + std::fmod(static_cast<double>(ii) + 1.0, 3000.0)
130                                 * 10.0) + 10.0;
131                         num += 10.0;
132                     }
133
134                     d = rt_roundd_snf(sensorParam->scolor);
135                     if (d < 1.8446744073709552E+19) {
136                         if (d >= 0.0) {
137                             u = static_cast<unsigned long long>(d);
138                         } else {
139                             u = 0ULL;
140                         }
141                     } else if (d >= 1.8446744073709552E+19) {
142                         u = MAX_uint64_T;
143                     } else {
144                         u = 0ULL;
145                     }
146
147                     coveredMap[ii] = u;
148
149                     // Set the line points to white
150                     sum_y += std::floor((static_cast<double>(ii) + 1.0) / 3000.0);
151                     sum_x = (sum_x + std::fmod(static_cast<double>(ii) + 1.0, 3000.0))
152                         + 1.0;
153                     num++;
154                     ii++;
155                 }
156
157                 // 付žíř ſýſš
158                 // coveredMap(s) = coveredMap(s)+5;
159                 // num = num + 5;
160                 // phase = phase - theta*5;
161                 idx++;
162             }
163             } else {
164                 new_theta = rt_atan2d_snf(sum_y / num - position[1], sum_x / num -
165                     position[0]) - th;
166                 if (rtIsNaN(new_theta)) {
167                     // disp('error : phase is Nan, Dormammu!!');
168                     flag_ = 0.0;
169
170                     // phase = phase_;
171                 }
172
173                 // [phase sum_x/num-position(1), sum_y/num-position(2)]
174                 exitg2 = 1;
175             }
176         } while (exitg2 == 0);
177
178         // new_theta' žířú
179         if (rtIsNaN(new_theta)) {
180             int k;
181             unsigned int Cluster_num;
182
183             // žófa žž ſ
184             % % % % % % % % % % % % % % % % % % % % % %
185             position[0] = Node[0];
position[1] = Node[1];

```

```

186     cont = 0.0;
187     flag = 100.0;
188     for (idx = 0; idx < 6000000; idx++) {
189         coveredgMap[idx] = save_map[idx];
190         u = save_map[idx];
191         label[idx] = save_map[idx];
192         if (save_map[idx] > 230ULL) {
193             u = 255ULL;
194             label[idx] = 255ULL;
195         }
196         if (u < 230ULL) {
197             label[idx] = 0ULL;
198         }
199     }
200 }
201
202 b_imclose(label, uv);
203 b_blabel(uv, b_label);
204 distance = b_label[0];
205 for (k = 0; k < 5999999; k++) {
206     d = b_label[k + 1];
207     if (distance < d) {
208         distance = d;
209     }
210 }
211
212 // Úřížříj
213 Cluster_num = 0U;
214 num = 9.0E+6;
215 i = static_cast<int>(distance);
216 if (0 <= i - 1) {
217     il = countN;
218 }
219
220 for (k = 0; k < i; k++) {
221     int i2;
222     int b_jj;
223     boolean_T exitg3;
224     int b_k;
225     for (i2 = 0; i2 < 6000000; i2++) {
226         x[i2] = (b_label[i2] == static_cast<double>(k) + 1.0);
227     }
228
229     idx = 0;
230     b_ii.set_size(6000000);
231     jj.set_size(6000000);
232     ii = 1;
233     b_jj = 1;
234     exitg3 = false;
235     while ((!exitg3) && (b_jj <= 2000)) {
236         boolean_T guard1;
237         guard1 = false;
238         if (x[(ii + 3000 * (b_jj - 1)) - 1]) {
239             idx++;
240             b_ii[idx - 1] = ii;
241             jj[idx - 1] = static_cast<short>(b_jj);
242             if (idx >= 6000000) {
243                 exitg3 = true;
244             } else {
245                 guard1 = true;
246             }
247         } else {
248             guard1 = true;
249         }
250
251         if (guard1) {
252             ii++;
253             if (ii > 3000) {
254                 ii = 1;
255                 b_jj++;
256             }
257         }
258     }
259
260     if (1 > idx) {
261         i2 = 0;
262     } else {
263         i2 = idx;
264     }
265
266     b_ii.set_size(i2);
267     if (1 > idx) {
268         idx = 0;
269     }
270
271     jj.set_size(idx);
272

```

```

273     // [x y];
274     result.set_size(b_ii.size(0), 2);
275     idx = b_ii.size(0);
276     for (i2 = 0; i2 < idx; i2++) {
277         result[i2] = b_ii[i2];
278     }
279
280     idx = jj.size(0);
281     for (i2 = 0; i2 < idx; i2++) {
282         result[i2 + result.size(0)] = jj[i2];
283     }
284
285     idx = b_ii.size(0);
286     if (idx <= 2) {
287         idx = 2;
288     }
289
290     if (b_ii.size(0) == 0) {
291         idx = 0;
292     }
293
294     ii = result.size(0);
295     if (result.size(0) == 0) {
296         Uncovered[0] = 0.0;
297         Uncovered[1] = 0.0;
298     } else {
299         Uncovered[0] = result[0];
300         for (b_k = 2; b_k <= ii; b_k++) {
301             Uncovered[0] += static_cast<double>(result[b_k - 1]);
302         }
303
304         Uncovered[1] = result[result.size(0)];
305         for (b_k = 2; b_k <= ii; b_k++) {
306             Uncovered[1] += static_cast<double>(result[(result.size(0) + b_k)
307                 - 1]);
308         }
309     }
310
311     Uncovered[0] = rt_roundd_snf(Uncovered[0] / static_cast<double>(idx));
312     Uncovered[1] = rt_roundd_snf(Uncovered[1] / static_cast<double>(idx));
313
314     // a
315     for (b_jj = 0; b_jj <= i1; b_jj++) {
316         i2 = b_jj << 1;
317         d = Uncovered[0] - Tree[i2];
318         distance = d * d;
319         idx = i2 + 1;
320         d = Uncovered[1] - Tree[idx];
321         distance += d * d;
322
323         // distance žíjí f
324         if (num > distance) {
325             kw_point2ind(Node, Uncovered, index2);
326             ii = 1;
327             b_k = 0;
328             exitg3 = false;
329             while ((!exitg3) && (b_k <= index2.size(1) - 1)) {
330                 if (binaryMap[static_cast<int>(index2[b_k]) - 1] < 100ULL) {
331                     // 特žří ŷýš
332                     ii = 0;
333                     exitg3 = true;
334                 } else {
335                     b_k++;
336                 }
337             }
338
339             if (ii != 0) {
340                 // ře ij
341                 num = distance;
342                 Cluster_num = k + 1U;
343                 position[0] = Tree[i2];
344                 Cluster_idx_0 = Uncovered[0];
345                 position[1] = Tree[idx];
346                 Cluster_idx_1 = Uncovered[1];
347
348                 // cellšn ■ aýù
349             }
350         }
351     }
352 }
353
354 if (Cluster_num == 0U) {
355     exitg1 = 1;
356 } else {
357     // distance žíjí f
358     th = rt_atan2d_snf(Cluster_idx_1 - position[1], Cluster_idx_0 -
359                         position[0]);

```

```

360
361         // ź
362         //      title("The Robot number is " + robot,'FontSize', 22);
363         //      imshow(BW);
364     }
365 } else {
366     // źœa ¿ž ź
367     % % % % % % % % % % % % % % % % % % % % % % % % % % %
368     d = std::abs(new_theta);
369     if (d > 180.0) {
370         new_theta -= 360.0 * new_theta / d;
371     }
372     th += 2.0 * new_theta;
373
374     // GAIN
375     if (rtIsInf(cont)) {
376         d = rtNaN;
377     } else {
378         d = std::fmod(cont, 20.0);
379     }
380
381     if (d == 0.0) {
382         // 20ž fÿüt'1Node
383         Node[0] = position[0];
384         Node[1] = position[1];
385         for (i = 0; i < 31; i++) {
386             unsigned short u1;
387             d = rt_rounnd_snf(position[1] + (static_cast<double>(i) + -16.0));
388             if (d < 65536.0) {
389                 if (d >= 0.0) {
390                     u1 = static_cast<unsigned short>(d);
391                 } else {
392                     u1 = 0U;
393                 }
394             } else if (d >= 65536.0) {
395                 u1 = MAX_uint16_T;
396             } else {
397                 u1 = 0U;
398             }
399
400             for (int i2 = 0; i2 < 31; i2++) {
401                 unsigned int Cluster_num;
402                 unsigned short u2;
403                 unsigned int u3;
404                 d = rt_rounnd_snf(position[0] + (static_cast<double>(i2) + -16.0));
405                 if (d < 65536.0) {
406                     if (d >= 0.0) {
407                         u2 = static_cast<unsigned short>(d);
408                     } else {
409                         u2 = 0U;
410                     }
411                 } else if (d >= 65536.0) {
412                     u2 = MAX_uint16_T;
413                 } else {
414                     u2 = 0U;
415                 }
416
417                 Cluster_num = u2 + 1U;
418                 if (Cluster_num > 65535U) {
419                     Cluster_num = 65535U;
420                 }
421
422                 u3 = u1 + 1U;
423                 if (u3 > 65535U) {
424                     u3 = 65535U;
425                 }
426
427                 coveredMap[(static_cast<int>(Cluster_num) + 3000 * (static_cast<
428                     int>(u3) - 1)) - 1] = 170ULL;
429             }
430         }
431
432         idx = (countN + 1) « 1;
433         Tree[idx] = position[0];
434         Tree[idx + 1] = position[1];
435         countN++;
436         std::memcpy(&save_map[0], &coveredMap[0], 6000000U * sizeof(unsigned
437             long long));
438     }
439
440     if (flag_ > 0.0) {
441         flag = 30.0;
442
443         // 30 ž
444     } else {
445         flag--;

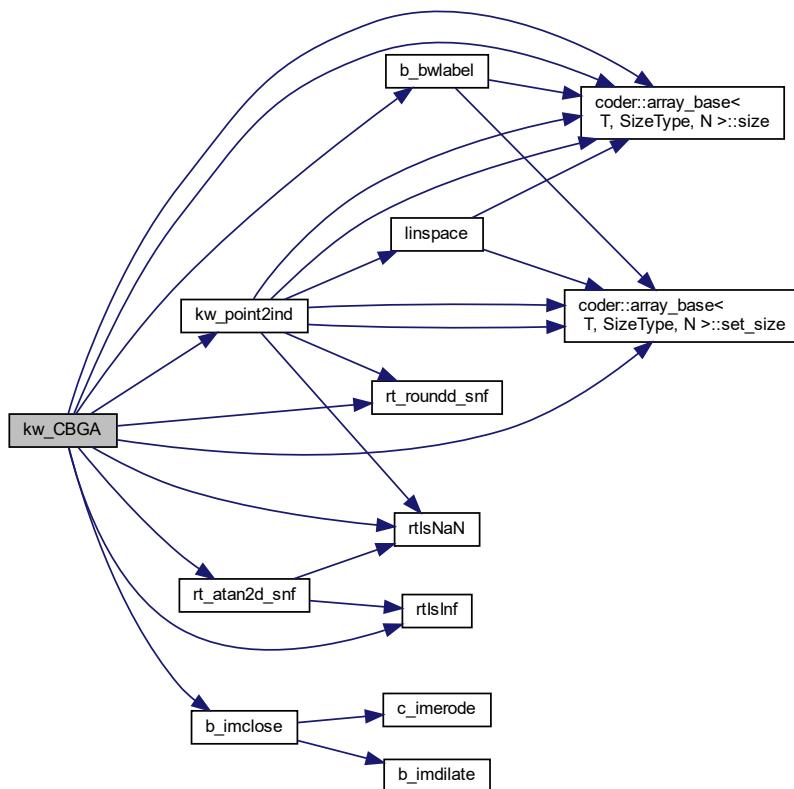
```

```

446      }
447      // ˇ
448      //      title("The Robot number is " + robot,'FontSize', 22);
449      //      imshow(BW);
450    }
451  } else {
452   std::memcpy(&coveredMap[0], &save_map[0], 6000000U * sizeof(unsigned
453   long long));
454
455   // 30 Ÿf > 1šNodet' f俸;
456   for (int k = 0; k < 200; k++) {
457     Tree[k] = rt_roundd_snf(Tree[k]);
458   }
459
460   exitgl = 1;
461 }
462 } while (exitgl == 0);
463
464 // 3 789 4
465 // Tree = [Tree(:,1:end-1), fliplr(Tree(:,1:end-2))];           % 3 78987 34
466
467 }

```

Here is the call graph for this function:

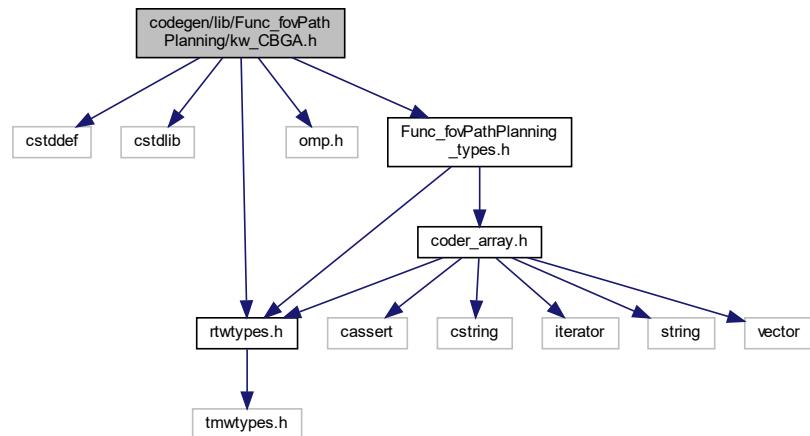


Here is the caller graph for this function:

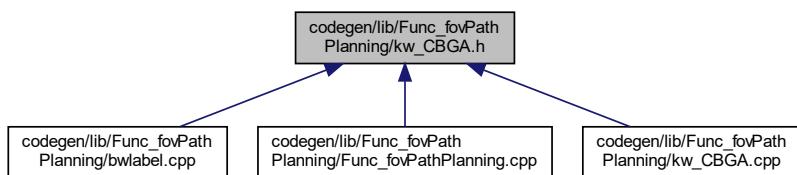


7.38 codegen/lib(Func_fovPathPlanning/kw_CBGA.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for kw_CBGA.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void kw_CBGA (struct_T *sensorParam, unsigned long long coveredMap[6000000], const unsigned long long binaryMap[6000000], double position[2], double th, double Tree[200])`

7.38.1 Macro Definition Documentation

7.38.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file kw_CBGA.h.

7.38.2 Function Documentation

7.38.2.1 kw_CBGA()

```
void kw_CBGA (
    struct_T * sensorParam,
    unsigned long long coveredMap[6000000],
    const unsigned long long binaryMap[6000000],
    double position[2],
    double th,
    double Tree[200] )
```

Definition at line 26 of file kw_CBGA.cpp.

```
29 {
30     double flag;
31     double cont;
32     double Node[2];
33     static unsigned long long save_map[6000000];
34     int countN;
35     double Cluster_idx_0;
36     double Cluster_idx_1;
37     double Uncovered[2];
38     double new_theta;
39     double num;
40     double sum_y;
41     double distance;
42     coder::array<double, 2U> index2;
43     static unsigned long long label[6000000];
44     static unsigned long long uv[6000000];
45     static double b_label[6000000];
46     int il;
47     static boolean_T x[6000000];
48     coder::array<int, 1U> b_ii;
49     coder::array<short, 1U> jj;
50     coder::array<int, 2U> result;
51     flag = 30.0;
52     cont = 0.0;
```

```

53     Node[0] = position[0];
54     Node[1] = position[1];
55     std::memcpy(&save_map[0], &coveredMap[0], 6000000U * sizeof(unsigned long
56     long));
57
58     // position = position + (sensorParam.dvec*3)*[cos(th), sin(th)];
59     std::memset(&Tree[0], 0, 200U * sizeof(double));
60
61     //      nearCellcovered tříš
62     countN = -1;
63     Tree[0] = position[0];
64     Cluster_idx_0 = 0.0;
65     Tree[1] = position[1];
66     Cluster_idx_1 = 0.0;
67     int exitg1;
68     do {
69         exitg1 = 0;
70         if (flag != 0.0) {
71             double sum_x;
72             double flag_;
73             int i;
74             int idx;
75             int ii;
76             double d;
77             unsigned long long u;
78             cont++;
79             Uncovered[0] = position[0] + 5.0 * std::cos(th);
80             Uncovered[1] = position[1] + 5.0 * std::sin(th);
81             position[0] = Uncovered[0];
82             position[1] = Uncovered[1];
83             new_theta = th;
84
85             // tříšyúzí gğ ijd'
86             //      if( frame(position(1), position(2))==0 )
87             //          phase = -phase_;
88             //      return;
89             //      end
90             sensorParam->scolor += sensorParam->dc;
91             if (sensorParam->scolor > 150.0) {
92                 sensorParam->dc = -1.0;
93             } else {
94                 if (sensorParam->scolor < 70.0) {
95                     sensorParam->dc = 1.0;
96                 }
97             }
98
99             num = 0.0;
100            sum_y = 0.0;
101            sum_x = 0.0;
102            flag_ = 0.0;
103            i = static_cast<int>(((th + 0.6108652381980153) + (0.0043633231299858239 -
104            (th - 0.6108652381980153)) / 0.0043633231299858239);
105            idx = 0;
106            int exitg2;
107            do {
108                exitg2 = 0;
109                if (idx <= i - 1) {
110                    distance = (th - 0.6108652381980153) + static_cast<double>(idx) *
111                        0.0043633231299858239;
112                    Uncovered[0] = position[0] + 300.0 * std::cos(distance);
113                    Uncovered[1] = position[1] + 300.0 * std::sin(distance);
114                    kw_point2ind(position, Uncovered, index2);
115
116                    // Linear indices
117                    if (index2.size(1) == 0) {
118                        flag_ = 0.0;
119                        exitg2 = 1;
120                    } else {
121                        ii = 0;
122                        while ((ii <= static_cast<int>(index2[0]) - 1) && (binaryMap[ii] >=
123                            200ULL)) {
124                            // flag = flag + (coveredMap(s) > 230);
125                            if (coveredMap[ii] > 230ULL) {
126                                flag_++;
127                                sum_y += std::floor((static_cast<double>(ii) + 1.0) / 3000.0) *
128                                    10.0;
129                                sum_x = (sum_x + std::fmod(static_cast<double>(ii) + 1.0, 3000.0)
130                                    * 10.0) + 10.0;
131                                num += 10.0;
132                            }
133
134                            d = rt_roundd_snf(sensorParam->scolor);
135                            if (d < 1.8446744073709552E+19) {
136                                if (d >= 0.0) {
137                                    u = static_cast<unsigned long long>(d);
138                                } else {
139                                    u = 0ULL;

```

```

140         }
141     } else if (d >= 1.8446744073709552E+19) {
142     u = MAX_uint64_T;
143     } else {
144     u = OULL;
145     }
146
147     coveredMap[ii] = u;
148
149     // Set the line points to white
150     sum_y += std::floor((static_cast<double>(ii) + 1.0) / 3000.0);
151     sum_x = (sum_x + std::fmod(static_cast<double>(ii) + 1.0, 3000.0))
152         + 1.0;
153     num++;
154     ii++;
155 }
156
157     // 特íří ſýšš
158     // coveredMap(s) = coveredMap(s)+5;
159     // num = num + 5;
160     // phase = phase - theta*5;
161     idx++;
162 }
163 else {
164     new_theta = rt_atan2d_snf(sum_y / num - position[1], sum_x / num -
165         position[0]) - th;
166     if (rtIsNaN(new_theta)) {
167         // disp('error : phase is Nan, Dormammu!!');
168         flag_ = 0.0;
169
170         // phase = phase_;
171     }
172
173     // [phase sum_x/num-position(1), sum_y/num-position(2)]
174     exitg2 = 1;
175 }
176 } while (exitg2 == 0);
177
178 // new_theta' říđó
179 if (rtIsNaN(new_theta)) {
180     int k;
181     unsigned int Cluster_num;
182
183     // Žófa źz ř
184     % % % % % % % % % % % % % % % % % % % % % % % % % % %
185     position[0] = Node[0];
186     position[1] = Node[1];
187     cont = 0.0;
188     flag = 100.0;
189     for (idx = 0; idx < 6000000; idx++) {
190         coveredMap[idx] = save_map[idx];
191         u = save_map[idx];
192         label[idx] = save_map[idx];
193         if (save_map[idx] > 230ULL) {
194             u = 255ULL;
195             label[idx] = 255ULL;
196         }
197         if (u < 230ULL) {
198             label[idx] = OULL;
199         }
200     }
201
202     b_imclose(label, uv);
203     b_bwlabel(uv, b_label);
204     distance = b_label[0];
205     for (k = 0; k < 5999999; k++) {
206         d = b_label[k + 1];
207         if (distance < d) {
208             distance = d;
209         }
210     }
211
212     // Úří;žříj
213     Cluster_num = 0U;
214     num = 9.0E+6;
215     i = static_cast<int>(distance);
216     if (0 <= i - 1) {
217         il = countN;
218     }
219
220     for (k = 0; k < i; k++) {
221         int i2;
222         int b_jj;
223         boolean_T exitg3;
224         int b_k;
225         for (i2 = 0; i2 < 6000000; i2++) {

```

```

226         x[i2] = (b_label[i2] == static_cast<double>(k) + 1.0);
227     }
228
229     idx = 0;
230     b_ii.set_size(6000000);
231     jj.set_size(6000000);
232     ii = 1;
233     b_jj = 1;
234     exitg3 = false;
235     while ((!exitg3) && (b_jj <= 2000)) {
236         boolean_T guard1 = false;
237         guard1 = false;
238         if (x[(ii + 3000 * (b_jj - 1)) - 1]) {
239             idx++;
240             b_ii[idx - 1] = ii;
241             jj[idx - 1] = static_cast<short>(b_jj);
242             if (idx >= 6000000) {
243                 exitg3 = true;
244             } else {
245                 guard1 = true;
246             }
247             } else {
248                 guard1 = true;
249             }
250
251         if (guard1) {
252             ii++;
253             if (ii > 3000) {
254                 ii = 1;
255                 b_jj++;
256             }
257         }
258     }
259
260     if (1 > idx) {
261         i2 = 0;
262     } else {
263         i2 = idx;
264     }
265
266     b_ii.set_size(i2);
267     if (1 > idx) {
268         idx = 0;
269     }
270
271     jj.set_size(idx);
272
273     // [x y];
274     result.set_size(b_ii.size(0), 2);
275     idx = b_ii.size(0);
276     for (i2 = 0; i2 < idx; i2++) {
277         result[i2] = b_ii[i2];
278     }
279
280     idx = jj.size(0);
281     for (i2 = 0; i2 < idx; i2++) {
282         result[i2 + result.size(0)] = jj[i2];
283     }
284
285     idx = b_ii.size(0);
286     if (idx <= 2) {
287         idx = 2;
288     }
289
290     if (b_ii.size(0) == 0) {
291         idx = 0;
292     }
293
294     ii = result.size(0);
295     if (result.size(0) == 0) {
296         Uncovered[0] = 0.0;
297         Uncovered[1] = 0.0;
298     } else {
299         Uncovered[0] = result[0];
300         for (b_k = 2; b_k <= ii; b_k++) {
301             Uncovered[0] += static_cast<double>(result[b_k - 1]);
302         }
303
304         Uncovered[1] = result[result.size(0)];
305         for (b_k = 2; b_k <= ii; b_k++) {
306             Uncovered[1] += static_cast<double>(result[(result.size(0) + b_k)
307                 - 1]);
308         }
309     }
310
311     Uncovered[0] = rt_roundd_snf(Uncovered[0] / static_cast<double>(idx));
312     Uncovered[1] = rt_roundd_snf(Uncovered[1] / static_cast<double>(idx));

```

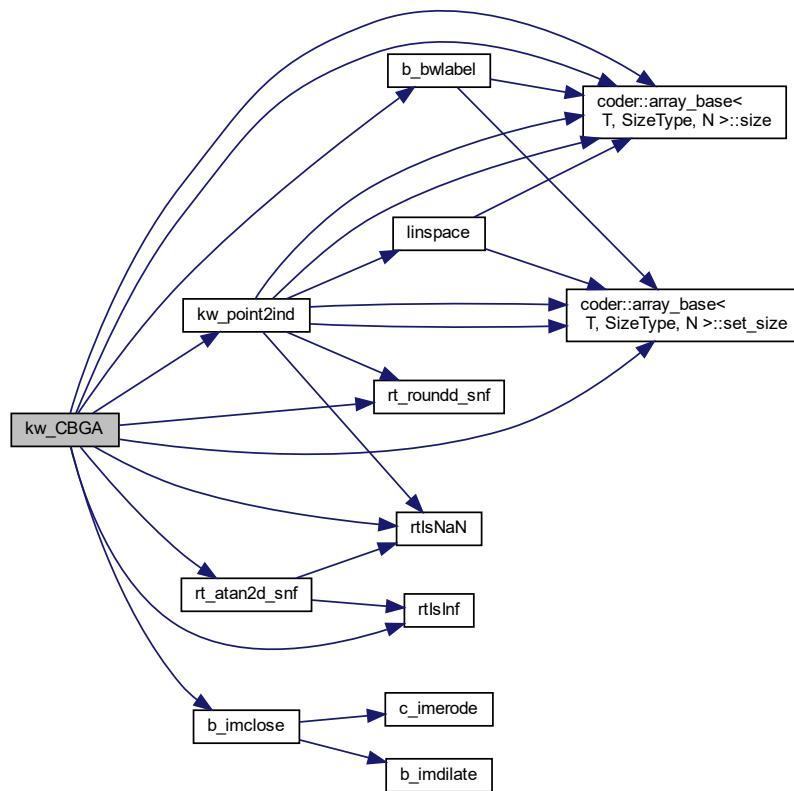
```

313     // a
314     for (b_jj = 0; b_jj <= i1; b_jj++) {
315         i2 = b_jj « 1;
316         d = Uncovered[0] - Tree[i2];
317         distance = d * d;
318         idx = i2 + 1;
319         d = Uncovered[1] - Tree[idx];
320         distance += d * d;
321
322         // distance žříj f
323         if (num > distance) {
324             kw_point2ind(Node, Uncovered, index2);
325             ii = 1;
326             b_k = 0;
327             exitg3 = false;
328             while ((!exitg3) && (b_k <= index2.size(1) - 1)) {
329                 if (binaryMap[static_cast<int>(index2[b_k]) - 1] < 100ULL) {
330                     // tříž ſyšš
331                     ii = 0;
332                     exitg3 = true;
333                 } else {
334                     b_k++;
335                 }
336             }
337         }
338
339         if (ii != 0) {
340             // ře ij
341             num = distance;
342             Cluster_num = k + 1U;
343             position[0] = Tree[i2];
344             Cluster_idx_0 = Uncovered[0];
345             position[1] = Tree[idx];
346             Cluster_idx_1 = Uncovered[1];
347
348             // cellšní ■ a jyù
349         }
350     }
351 }
352
353 if (Cluster_num == 0U) {
354     exitgl = 1;
355 } else {
356     // distance žříj f
357     th = rt_atan2d_snf(Cluster_idx_1 - position[1], Cluster_idx_0 -
358     position[0]);
359
360     //
361     // title("The Robot number is " + robot,'FontSize', 22);
362     // imshow(BW);
363 }
364
365 } else {
366     // žofa, ž ř
367     % % % % % % % % % % % % % % % % % % % % % % % % % % %
368     d = std::abs(new_theta);
369     if (d > 180.0) {
370         new_theta -= 360.0 * new_theta / d;
371     }
372
373     th += 2.0 * new_theta;
374
375     // GAIN
376     if (rtIsInf(cont)) {
377         d = rtNaN;
378     } else {
379         d = std::fmod(cont, 20.0);
380     }
381
382     if (d == 0.0) {
383         // 20z ſyút'1Node
384         Node[0] = position[0];
385         Node[1] = position[1];
386         for (i = 0; i < 31; i++) {
387             unsigned short ul;
388             d = rt_roundd_snf(position[1] + (static_cast<double>(i) + -16.0));
389             if (d < 65536.0) {
390                 if (d >= 0.0) {
391                     ul = static_cast<unsigned short>(d);
392                 } else {
393                     ul = 0U;
394                 }
395             } else if (d >= 65536.0) {
396                 ul = MAX_uint16_T;
397             } else {
398                 ul = 0U;
399             }
400         }
401     }
402 }
```

```

399
400     for (int i2 = 0; i2 < 31; i2++) {
401         unsigned int Cluster_num;
402         unsigned short u2;
403         unsigned int u3;
404         d = rt_round_snf(position[0] + (static_cast<double>(i2) + -16.0));
405         if (d < 65536.0) {
406             if (d >= 0.0) {
407                 u2 = static_cast<unsigned short>(d);
408             } else {
409                 u2 = 0U;
410             }
411         } else if (d >= 65536.0) {
412             u2 = MAX_uint16_T;
413         } else {
414             u2 = 0U;
415         }
416
417         Cluster_num = u2 + 1U;
418         if (Cluster_num > 65535U) {
419             Cluster_num = 65535U;
420         }
421
422         u3 = u1 + 1U;
423         if (u3 > 65535U) {
424             u3 = 65535U;
425         }
426
427         coveredMap[(static_cast<int>(Cluster_num) + 3000 * (static_cast<
428             int>(u3) - 1)) - 1] = 170ULL;
429     }
430 }
431
432     idx = (countN + 1) << 1;
433     Tree[idx] = position[0];
434     Tree[idx + 1] = position[1];
435     countN++;
436     std::memcpy(&save_map[0], &coveredMap[0], 6000000U * sizeof(unsigned
437         long long));
438 }
439
440     if (flag_ > 0.0) {
441         flag = 30.0;
442
443         // 30 żf
444     } else {
445         flag--;
446     }
447
448     // ź
449     // title("The Robot number is " + robot,'FontSize', 22);
450     // imshow(BW);
451 }
452 } else {
453     std::memcpy(&coveredMap[0], &save_map[0], 6000000U * sizeof(unsigned
454         long long));
455
456     // 30 żf > 1şNodet' f
457     for (int k = 0; k < 200; k++) {
458         Tree[k] = rt_round_snf(Tree[k]);
459     }
460
461     exitg1 = 1;
462 }
463 } while (exitg1 == 0);
464
465 // 3 789 4
466 // Tree = [Tree(:,1:end-1), fliplr(Tree(:,1:end-2))];           % 3 78987 34
467 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



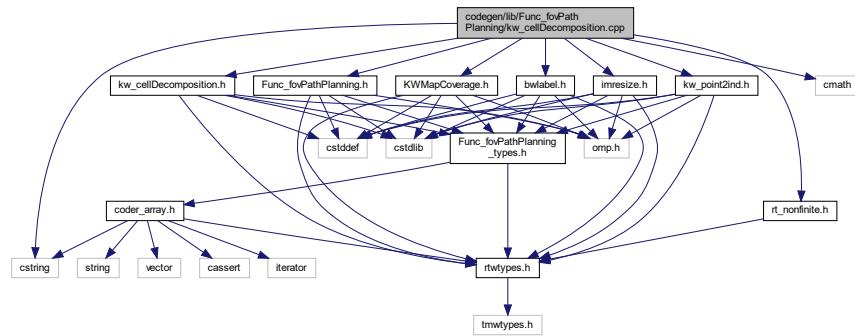
7.39 codegen/lib/Func_fovPathPlanning/kw_cellDecomposition.cpp File Reference

```

#include "kw_cellDecomposition.h"
#include "Func_fovPathPlanning.h"
#include "KWMapCoverage.h"
#include "bwlabel.h"
#include "imresize.h"
#include "kw_point2ind.h"

```

```
#include "rt_nonfinite.h"
#include <cmath>
#include <cstring>
Include dependency graph for kw_cellDecomposition.cpp:
```



Functions

- double **b_kw_cellDecomposition** (const unsigned long long coveredMap[6000000], **cell_wrap_1** coordix[3], **cell_wrap_1** coordiy[3])
- void **kw_cellDecomposition** (const unsigned long long coveredMap[6000000], const **cell_wrap_0** coordix[3], const **cell_wrap_0** coordiy[3], **cell_wrap_1** cells[3], **cell_wrap_1** cell_index[3], double *numCluster)

7.39.1 Function Documentation

7.39.1.1 b_kw_cellDecomposition()

```
double b_kw_cellDecomposition (
    const unsigned long long coveredMap[6000000],
    cell_wrap_1 coordix[3],
    cell_wrap_1 coordiy[3] )
```

Definition at line 24 of file kw_cellDecomposition.cpp.

```
26 {
27     double numCluster;
28     static unsigned long long Mat[6000000];
29     int idx;
30     unsigned long long uv[2542];
31     int x;
32     double label[2542];
33     int i;
34     double ex;
35     short ii_data[2542];
36     short ss_data[2542];
37     coder::array<double, 2U> b_coordix;
38
39 //  imresize(imresize(kw_BWfilter(coveredMap),0.02,'nearest'),50,'nearest')
40 std::memcpy(&Mat[0], &coveredMap[0], 6000000U * sizeof(unsigned long long));
41 for (idx = 0; idx < 3000; idx++) {
42     for (x = 0; x < 2000; x++) {
43         i = idx + 3000 * x;
44         if (Mat[i] < 230ULL) {
45             Mat[i] = 0ULL;
46         } else {
47             Mat[i] = 255ULL;
```

```

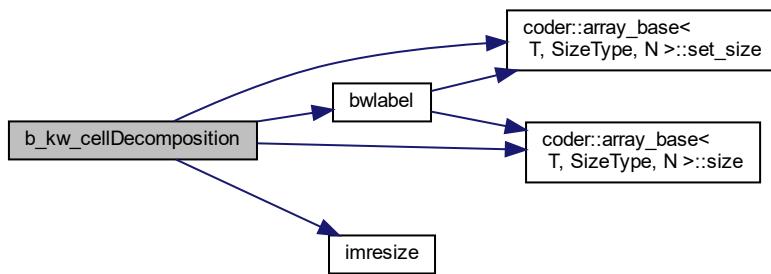
48         }
49     }
50 }
51
52 imresize(Mat, uv);
53 bwlabel(uv, label);
54
55 // imresize(label,0.02,'nearest');
56 ex = label[0];
57 for (idx = 0; idx < 2541; idx++) {
58     double d;
59     d = label[idx + 1];
60     if (ex < d) {
61         ex = d;
62     }
63 }
64
65 i = static_cast<int>(ex);
66 numCluster = 1.0;
67 for (int b_numCluster = 0; b_numCluster < i; b_numCluster++) {
68     boolean_T exitgl;
69     numCluster = static_cast<double>(b_numCluster) + 1.0;
70
71     // Úřížříj
72     coordix[b_numCluster].f1.set_size(0, 0);
73     coordiy[b_numCluster].f1.set_size(0, 0);
74     idx = 0;
75     x = 0;
76     exitgl = false;
77     while ((!exitgl) && (x < 2542)) {
78         if (label[x] == static_cast<double>(b_numCluster) + 1.0) {
79             idx++;
80             ii_data[idx - 1] = static_cast<short>(x + 1);
81             if (idx >= 2542) {
82                 exitgl = true;
83             } else {
84                 x++;
85             }
86         } else {
87             x++;
88         }
89     }
90
91     if (1 > idx) {
92         idx = 0;
93     }
94
95     if (0 <= idx - 1) {
96         std::memcpy(&ss_data[0], &ii_data[0], idx * sizeof(short));
97     }
98
99     idx = ss_data[0];
100    for (int s = 0; s < idx; s++) {
101        int y;
102        int loop_ub;
103        int il;
104        x = static_cast<int>(std::fmod(static_cast<double>(s) + 1.0, 62.0)) * 49 -
105            24;
106        y = static_cast<int>(std::floor((static_cast<double>(s) + 1.0) / 62.0)) *
107            49 + 23;
108        if ((coordix[b_numCluster].f1.size(0) != 0) && (coordix[b_numCluster].
109            f1.size(1) != 0)) {
110            loop_ub = coordix[b_numCluster].f1.size(1);
111        } else {
112            loop_ub = 0;
113        }
114
115        b_coordix.set_size(1, (loop_ub + 2));
116        for (il = 0; il < loop_ub; il++) {
117            b_coordix[b_coordix.size(0) * il] = coordix[b_numCluster].f1[il];
118        }
119
120        b_coordix[b_coordix.size(0) * loop_ub] = x;
121        b_coordix[b_coordix.size(0) * (loop_ub + 1)] = static_cast<double>(y) +
122            1.0;
123        coordix[b_numCluster].f1.set_size(b_coordix.size(0), b_coordix.size(1));
124        loop_ub = b_coordix.size(0) * b_coordix.size(1);
125        for (il = 0; il < loop_ub; il++) {
126            coordix[b_numCluster].f1[il] = b_coordix[il];
127        }
128
129        if ((coordiy[b_numCluster].f1.size(0) != 0) && (coordiy[b_numCluster].
130            f1.size(1) != 0)) {
131            loop_ub = coordiy[b_numCluster].f1.size(1);
132        } else {
133            loop_ub = 0;
134        }

```

```

135
136     b_coordix.set_size(1, (loop_ub + 1));
137     for (il = 0; il < loop_ub; il++) {
138         b_coordix[b_coordix.size(0) * il] = coordiy[b_numCluster].f1[il];
139     }
140
141     b_coordix[b_coordix.size(0) * loop_ub] = x + 3000 * y;
142     coordiy[b_numCluster].f1.set_size(b_coordix.size(0), b_coordix.size(1));
143     loop_ub = b_coordix.size(0) * b_coordix.size(1);
144     for (il = 0; il < loop_ub; il++) {
145         coordiy[b_numCluster].f1[il] = b_coordix[il];
146     }
147 }
148 }
149
150 // %
151 return numCluster;
152 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.39.1.2 kw_cellDecomposition()

```

void kw_cellDecomposition (
    const unsigned long long coveredMap[6000000],
    const cell_wrap_0 coordix[3],
    const cell_wrap_0 coordiy[3],
    cell_wrap_1 cells[3],
    cell_wrap_1 cell_index[3],
    double * numCluster )
```

Definition at line 154 of file kw_cellDecomposition.cpp.

```

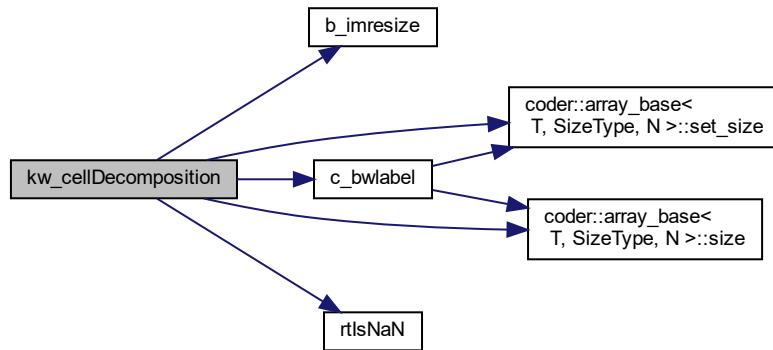
157 {
158     static unsigned long long Mat[6000000];
159     int idx;
160     unsigned long long tmp_data[3844];
161     int tmp_size[2];
162     int j;
163     double label_data[3844];
164     int label_size[2];
165     int H;
166     int i;
167     double x;
168     double y;
169     unsigned int b_numCluster;
170     int loop_ub;
171     int nx;
172     boolean_T x_data[3844];
173     short ii_data[3844];
174     short ss_data[3844];
175     coder::array<double, 2U> b_cells;
176
177 // imresize(imresize(kw_BWfilter(coveredMap),0.02,'nearest'),50,'nearest')
178 std::memcpy(&Mat[0], &coveredMap[0], 6000000U * sizeof(unsigned long long));
179 for (idx = 0; idx < 3000; idx++) {
180     for (j = 0; j < 2000; j++) {
181         i = idx + 3000 * j;
182         if (Mat[i] < 230ULL) {
183             Mat[i] = 0ULL;
184         } else {
185             Mat[i] = 255ULL;
186         }
187     }
188 }
189 b_imresize(Mat, tmp_data, tmp_size);
190 c_bwlabel(tmp_data, tmp_size, label_data, label_size);
H = label_size[0];
193 for (i = 0; i < 3; i++) {
194     cells[i].f1.set_size(1, 10);
195     cell_index[i].f1.set_size(1, 10);
196     for (idx = 0; idx < 10; idx++) {
197         cells[i].f1[idx] = coordix[i].f1[idx];
198         cell_index[i].f1[idx] = coordiy[i].f1[idx];
199     }
200 }
201
202 // imresize(label,0.02,'nearest');
203 idx = label_size[0] * label_size[1];
204 if (idx <= 2) {
205     if (idx == 1) {
206         x = label_data[0];
207     } else if ((label_data[0] < label_data[1]) || (rtIsNaN(label_data[0]) &&
208             (!rtIsNaN(label_data[1])))) {
209         x = label_data[1];
210     } else {
211         x = label_data[0];
212     }
213 } else {
214     x = label_data[0];
215     for (j = 2; j <= idx; j++) {
216         y = label_data[j - 1];
217         if (x < y) {
218             x = y;
219         }
220     }
221 }
222
223 i = static_cast<int>(x);
224 b_numCluster = 1U;
225 if (0 <= i - 1) {
226     loop_ub = label_size[0] * label_size[1];
227     nx = label_size[0] * label_size[1];
228 }
229
230 for (int c_numCluster = 0; c_numCluster < i; c_numCluster++) {
231     int ii;
232     boolean_T exitg1;
233     b_numCluster = c_numCluster + 1U;
234
235 // Úřížíj
236     cells[c_numCluster].f1.set_size(0, 0);
237     cell_index[c_numCluster].f1.set_size(0, 0);
238     for (idx = 0; idx < loop_ub; idx++) {
239         x_data[idx] = (label_data[idx] == static_cast<double>(c_numCluster) + 1.0);
240     }
241     idx = 0;

```

```

243     j = nx;
244     ii = 0;
245     exitg1 = false;
246     while ((!exitg1) && (ii <= nx - 1)) {
247         if (x_data[ii]) {
248             idx++;
249             i_i_data[idx - 1] = static_cast<short>(ii + 1);
250             if (idx >= nx) {
251                 exitg1 = true;
252             } else {
253                 ii++;
254             }
255         } else {
256             ii++;
257         }
258     }
259
260     if (nx == 1) {
261         if (idx == 0) {
262             j = 0;
263         }
264         else if (1 > idx) {
265             j = 0;
266         } else {
267             j = idx;
268         }
269
270         if (0 <= j - 1) {
271             std::memcpy(&ss_data[0], &i_i_data[0], j * sizeof(short));
272         }
273
274         idx = ss_data[0];
275         for (ii = 0; ii < idx; ii++) {
276             int il;
277             x = std::fmod(static_cast<double>(ii) + 1.0, static_cast<double>(H)) *
278                 49.0 - 24.0;
279             y = std::floor((static_cast<double>(ii) + 1.0) / static_cast<double>(H)) *
280                 49.0 + 24.0;
281             if ((cells[c_numCluster].f1.size(0) != 0) && (cells[c_numCluster].f1.size
282                   (1) != 0)) {
283                 j = cells[c_numCluster].f1.size(1);
284             } else {
285                 j = 0;
286             }
287
288             b_cells.set_size(1, (j + 2));
289             for (il = 0; il < j; il++) {
290                 b_cells[b_cells.size(0) * il] = cells[c_numCluster].f1[il];
291             }
292
293             b_cells[b_cells.size(0) * j] = x;
294             b_cells[b_cells.size(0) * (j + 1)] = y;
295             cells[c_numCluster].f1.set_size(b_cells.size(0), b_cells.size(1));
296             j = b_cells.size(0) * b_cells.size(1);
297             for (il = 0; il < j; il++) {
298                 cells[c_numCluster].f1[il] = b_cells[il];
299             }
300
301             if ((cell_index[c_numCluster].f1.size(0) != 0) && (cell_index[c_numCluster]
302                   .f1.size(1) != 0)) {
303                 j = cell_index[c_numCluster].f1.size(1);
304             } else {
305                 j = 0;
306             }
307
308             b_cells.set_size(1, (j + 1));
309             for (il = 0; il < j; il++) {
310                 b_cells[b_cells.size(0) * il] = cell_index[c_numCluster].f1[il];
311             }
312
313             b_cells[b_cells.size(0) * j] = static_cast<int>(x) + 3000 * (static_cast<
314                 int>(y) - 1);
315             cell_index[c_numCluster].f1.set_size(b_cells.size(0), b_cells.size(1));
316             j = b_cells.size(0) * b_cells.size(1);
317             for (il = 0; il < j; il++) {
318                 cell_index[c_numCluster].f1[il] = b_cells[il];
319             }
320         }
321     }
322
323     //    %%
324     *numCluster = b_numCluster;
325 }
```

Here is the call graph for this function:



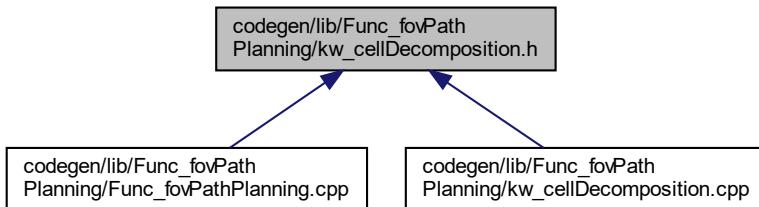
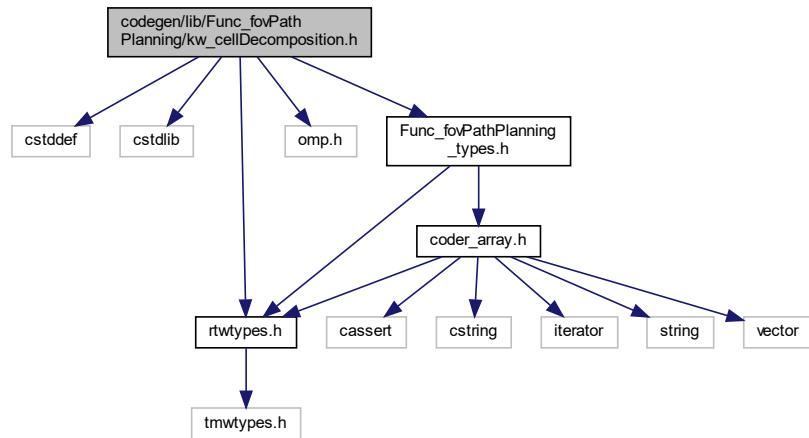
Here is the caller graph for this function:



7.40 codegen/lib/Func_fovPathPlanning/kw_cellDecomposition.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
```

Include dependency graph for kw_cellDecomposition.h:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `double b_kw_cellDecomposition (const unsigned long long coveredMap[6000000], cell_wrap_1 coordix[3], cell_wrap_1 coordiy[3])`
- `void kw_cellDecomposition (const unsigned long long coveredMap[6000000], const cell_wrap_0 coordix[3], const cell_wrap_0 coordiy[3], cell_wrap_1 cells[3], cell_wrap_1 cell_index[3], double *numCluster)`

7.40.1 Macro Definition Documentation

7.40.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file kw_cellDecomposition.h.

7.40.2 Function Documentation

7.40.2.1 b_kw_cellDecomposition()

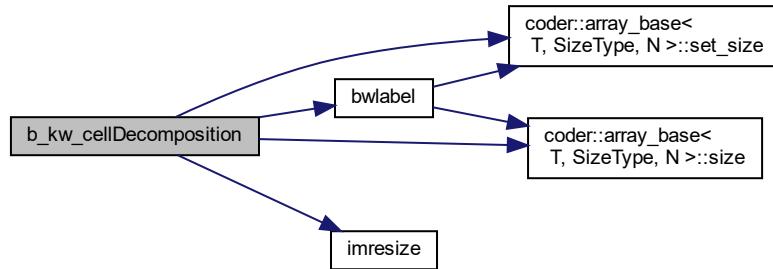
```
double b_kw_cellDecomposition (
        const unsigned long long coveredMap[6000000],
        cell_wrap_1 coordix[3],
        cell_wrap_1 coordiy[3] )
```

Definition at line 24 of file kw_cellDecomposition.cpp.

```
26 {
27     double numCluster;
28     static unsigned long long Mat[6000000];
29     int idx;
30     unsigned long long uv[2542];
31     int x;
32     double label[2542];
33     int i;
34     double ex;
35     short ii_data[2542];
36     short ss_data[2542];
37     coder::array<double, 2U> b_coordix;
38
39 //  imresize(imresize(kw_BWfilter(coveredMap),0.02,'nearest'),50,'nearest')
40 std::memcpy(&Mat[0], &coveredMap[0], 6000000U * sizeof(unsigned long long));
41 for (idx = 0; idx < 3000; idx++) {
42     for (x = 0; x < 2000; x++) {
43         i = idx + 3000 * x;
44         if (Mat[i] < 230ULL) {
45             Mat[i] = 0ULL;
46         } else {
47             Mat[i] = 255ULL;
48         }
49     }
50 }
51
52 imresize(Mat, uv);
53 bwlabel(uv, label);
54
55 // imresize(label,0.02,'nearest');
56 ex = label[0];
57 for (idx = 0; idx < 2541; idx++) {
58     double d;
59     d = label[idx + 1];
60     if (ex < d) {
61         ex = d;
62     }
63 }
64
65 i = static_cast<int>(ex);
66 numCluster = 1.0;
67 for (int b_numCluster = 0; b_numCluster < i; b_numCluster++) {
68     boolean_T exitg1;
69     numCluster = static_cast<double>(b_numCluster) + 1.0;
70
71 //  Ťříď
72 coordix[b_numCluster].f1.set_size(0, 0);
73 coordiy[b_numCluster].f1.set_size(0, 0);
74 idx = 0;
75 x = 0;
76 exitg1 = false;
77 while ((!exitg1) && (x < 2542)) {
78     if (label[x] == static_cast<double>(b_numCluster) + 1.0) {
79         idx++;
```

```
80         ii_data[idx - 1] = static_cast<short>(x + 1);
81         if (idx >= 2542) {
82             exitg1 = true;
83         } else {
84             x++;
85         }
86     } else {
87         x++;
88     }
89 }
90
91 if (l > idx) {
92     idx = 0;
93 }
94
95 if (0 <= idx - 1) {
96     std::memcpy(&ss_data[0], &ii_data[0], idx * sizeof(short));
97 }
98
99 idx = ss_data[0];
100 for (int s = 0; s < idx; s++) {
101     int y;
102     int loop_ub;
103     int il;
104     x = static_cast<int>(std::fmod(static_cast<double>(s) + 1.0, 62.0)) * 49 -
105         24;
106     y = static_cast<int>(std::floor((static_cast<double>(s) + 1.0) / 62.0)) *
107         49 + 23;
108     if ((coordix[b_numCluster].f1.size(0) != 0) && (coordix[b_numCluster].
109         f1.size(1) != 0)) {
110         loop_ub = coordix[b_numCluster].f1.size(1);
111     } else {
112         loop_ub = 0;
113     }
114
115     b_coordix.set_size(1, (loop_ub + 2));
116     for (il = 0; il < loop_ub; il++) {
117         b_coordix[b_coordix.size(0) * il] = coordix[b_numCluster].f1[il];
118     }
119
120     b_coordix[b_coordix.size(0) * loop_ub] = x;
121     b_coordix[b_coordix.size(0) * (loop_ub + 1)] = static_cast<double>(y) +
122         1.0;
123     coordix[b_numCluster].f1.set_size(b_coordix.size(0), b_coordix.size(1));
124     loop_ub = b_coordix.size(0) * b_coordix.size(1);
125     for (il = 0; il < loop_ub; il++) {
126         coordix[b_numCluster].f1[il] = b_coordix[il];
127     }
128
129     if ((coordiy[b_numCluster].f1.size(0) != 0) && (coordiy[b_numCluster].
130         f1.size(1) != 0)) {
131         loop_ub = coordiy[b_numCluster].f1.size(1);
132     } else {
133         loop_ub = 0;
134     }
135
136     b_coordix.set_size(1, (loop_ub + 1));
137     for (il = 0; il < loop_ub; il++) {
138         b_coordix[b_coordix.size(0) * il] = coordiy[b_numCluster].f1[il];
139     }
140
141     b_coordix[b_coordix.size(0) * loop_ub] = x + 3000 * y;
142     coordiy[b_numCluster].f1.set_size(b_coordix.size(0), b_coordix.size(1));
143     loop_ub = b_coordix.size(0) * b_coordix.size(1);
144     for (il = 0; il < loop_ub; il++) {
145         coordiy[b_numCluster].f1[il] = b_coordix[il];
146     }
147 }
148 }
149
150 //    %%
151 return numCluster;
152 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.40.2.2 kw_cellDecomposition()

```

void kw_cellDecomposition (
    const unsigned long long coveredMap[6000000],
    const cell_wrap_0 coordix[3],
    const cell_wrap_0 coordiy[3],
    cell_wrap_1 cells[3],
    cell_wrap_1 cell_index[3],
    double * numCluster )
  
```

Definition at line 154 of file `kw_cellDecomposition.cpp`.

```

157 {
158     static unsigned long long Mat[6000000];
159     int idx;
160     unsigned long long tmp_data[3844];
161     int tmp_size[2];
162     int j;
163     double label_data[3844];
164     int label_size[2];
165     int H;
166     int i;
167     double x;
168     double y;
169     unsigned int b_numCluster;
170     int loop_ub;
171     int nx;
172     boolean_T x_data[3844];
173     short ii_data[3844];
174     short ss_data[3844];
  
```

```

175  coder::array<double, 2U> b_cells;
176
177 // imresize(imresize(kw_BWfilter(coveredMap),0.02,'nearest'),50,'nearest')
178 std::memcpy(&Mat[0], &coveredMap[0], 6000000U * sizeof(unsigned long long));
179 for (idx = 0; idx < 3000; idx++) {
180   for (j = 0; j < 2000; j++) {
181     i = idx + 3000 * j;
182     if (Mat[i] < 230ULL) {
183       Mat[i] = 0ULL;
184     } else {
185       Mat[i] = 255ULL;
186     }
187   }
188 }
189
190 b_imresize(Mat, tmp_data, tmp_size);
191 c_bwlabel(tmp_data, tmp_size, label_data, label_size);
192 H = label_size[0];
193 for (i = 0; i < 3; i++) {
194   cells[i].f1.set_size(1, 10);
195   cell_index[i].f1.set_size(1, 10);
196   for (idx = 0; idx < 10; idx++) {
197     cells[i].f1[idx] = coordix[i].f1[idx];
198     cell_index[i].f1[idx] = coordiy[i].f1[idx];
199   }
200 }
201
202 // imresize(label,0.02,'nearest');
203 idx = label_size[0] * label_size[1];
204 if (idx <= 2) {
205   if (idx == 1) {
206     x = label_data[0];
207   } else if ((label_data[0] < label_data[1]) || (rtIsNaN(label_data[0]) &&
208             (!rtIsNaN(label_data[1])))) {
209     x = label_data[1];
210   } else {
211     x = label_data[0];
212   }
213 } else {
214   x = label_data[0];
215   for (j = 2; j <= idx; j++) {
216     y = label_data[j - 1];
217     if (x < y) {
218       x = y;
219     }
220   }
221 }
222
223 i = static_cast<int>(x);
224 b_numCluster = 1U;
225 if (0 <= i - 1) {
226   loop_ub = label_size[0] * label_size[1];
227   nx = label_size[0] * label_size[1];
228 }
229
230 for (int c_numCluster = 0; c_numCluster < i; c_numCluster++) {
231   int ii;
232   boolean_T exitg1;
233   b_numCluster = c_numCluster + 1U;
234
235 // Úřížříž
236 cells[c_numCluster].f1.set_size(0, 0);
237 cell_index[c_numCluster].f1.set_size(0, 0);
238 for (idx = 0; idx < loop_ub; idx++) {
239   x_data[idx] = (label_data[idx] == static_cast<double>(c_numCluster) + 1.0);
240 }
241
242 idx = 0;
243 j = nx;
244 ii = 0;
245 exitg1 = false;
246 while ((!exitg1) && (ii <= nx - 1)) {
247   if (x_data[ii]) {
248     idx++;
249     ii_data[idx - 1] = static_cast<short>(ii + 1);
250     if (idx >= nx) {
251       exitg1 = true;
252     } else {
253       ii++;
254     }
255   } else {
256     ii++;
257   }
258 }
259 if (nx == 1) {
260   if (idx == 0) {

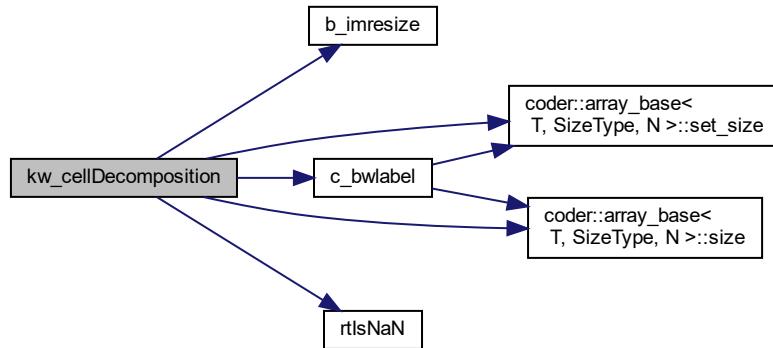
```

```

262         j = 0;
263     }
264 } else if (1 > idx) {
265     j = 0;
266 } else {
267     j = idx;
268 }
269
270 if (0 <= j - 1) {
271     std::memcpy(&ss_data[0], &ii_data[0], j * sizeof(short));
272 }
273
274 idx = ss_data[0];
275 for (ii = 0; ii < idx; ii++) {
276     int il;
277     x = std::fmod(static_cast<double>(ii) + 1.0, static_cast<double>(H)) *
278         49.0 - 24.0;
279     y = std::floor((static_cast<double>(ii) + 1.0) / static_cast<double>(H)) *
280         49.0 + 24.0;
281     if ((cells[c_numCluster].f1.size(0) != 0) && (cells[c_numCluster].f1.size
282         (1) != 0)) {
283         j = cells[c_numCluster].f1.size(1);
284     } else {
285         j = 0;
286     }
287
288 b_cells.set_size(1, (j + 2));
289 for (il = 0; il < j; il++) {
290     b_cells[b_cells.size(0) * il] = cells[c_numCluster].f1[il];
291 }
292
293 b_cells[b_cells.size(0) * j] = x;
294 b_cells[b_cells.size(0) * (j + 1)] = y;
295 cells[c_numCluster].f1.set_size(b_cells.size(0), b_cells.size(1));
296 j = b_cells.size(0) * b_cells.size(1);
297 for (il = 0; il < j; il++) {
298     cells[c_numCluster].f1[il] = b_cells[il];
299 }
300
301 if ((cell_index[c_numCluster].f1.size(0) != 0) && (cell_index[c_numCluster]
302     .f1.size(1) != 0)) {
303     j = cell_index[c_numCluster].f1.size(1);
304 } else {
305     j = 0;
306 }
307
308 b_cells.set_size(1, (j + 1));
309 for (il = 0; il < j; il++) {
310     b_cells[b_cells.size(0) * il] = cell_index[c_numCluster].f1[il];
311 }
312
313 b_cells[b_cells.size(0) * j] = static_cast<int>(x) + 3000 * (static_cast<
314     int>(y) - 1);
315 cell_index[c_numCluster].f1.set_size(b_cells.size(0), b_cells.size(1));
316 j = b_cells.size(0) * b_cells.size(1);
317 for (il = 0; il < j; il++) {
318     cell_index[c_numCluster].f1[il] = b_cells[il];
319 }
320 }
321 }
322
323 //    %%
324 *numCluster = b_numCluster;
325 }

```

Here is the call graph for this function:



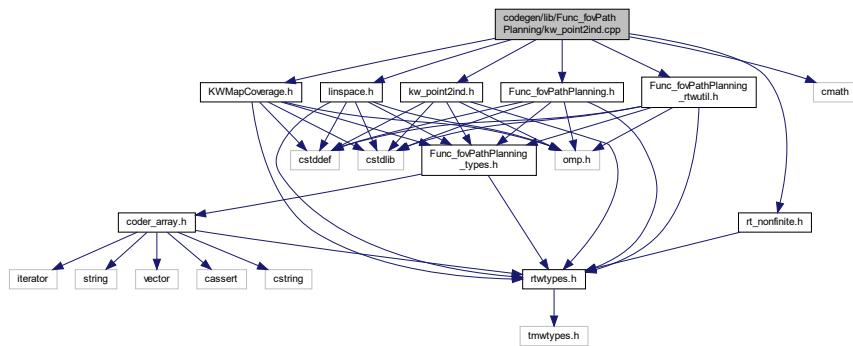
Here is the caller graph for this function:



7.41 codegen/lib/Func_fovPathPlanning/kw_point2ind.cpp File Reference

```
#include "kw_point2ind.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "KWMMapCoverage.h"
#include "linspace.h"
#include "rt_nonfinite.h"
#include <cmath>
```

Include dependency graph for kw_point2ind.cpp:



Functions

- void **kw_point2ind** (const double point1[2], const double point2[2], coder::array< double, 2U > &b_index)

7.41.1 Function Documentation

7.41.1.1 kw_point2ind()

```
void kw_point2ind (
    const double point1[2],
    const double point2[2],
    coder::array< double, 2U > & b_index )
```

Definition at line 22 of file kw_point2ind.cpp.

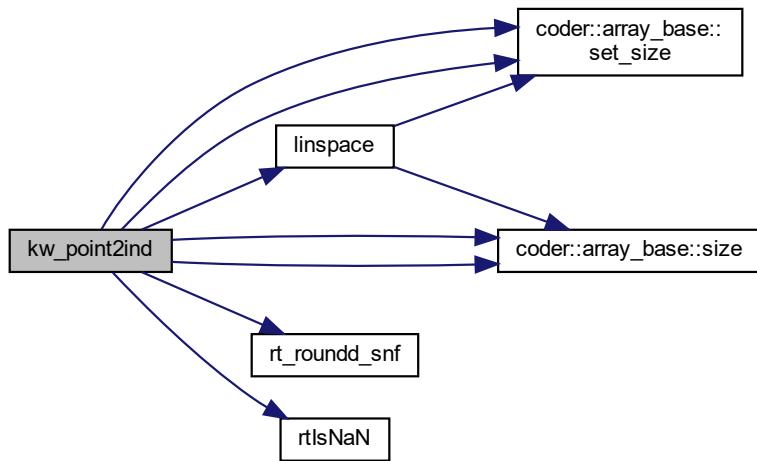
```

24 {
25     double u0;
26     double u1;
27     coder::array<double, 2U> rIndex;
28     int nx;
29     int n;
30     coder::array<double, 2U> cIndex;
31     coder::array<boolean_T, 2U> r;
32     coder::array<boolean_T, 2U> rl;
33     int trueCount;
34     int i;
35     u0 = std::abs(point2[0] - point1[0]);
36     u1 = std::abs(point2[1] - point1[1]);
37     if ((!(u0 > u1)) && (!rtIsNaN(u1))) {
38         u0 = u1;
39     }
40
41 // Number of points in line
42 linspace(point1[0], point2[0], u0 + 1.0, rIndex);
43 nx = rIndex.size(1);
44 for (n = 0; n < nx; n++) {
45     rIndex[n] = rt_roundd_snf(rIndex[n]);
46 }
47
48 // Row indices
49 linspace(point1[1], point2[1], u0 + 1.0, cIndex);
50 nx = cIndex.size(1);
51 for (n = 0; n < nx; n++) {
52     cIndex[n] = rt_roundd_snf(cIndex[n]);
53 }

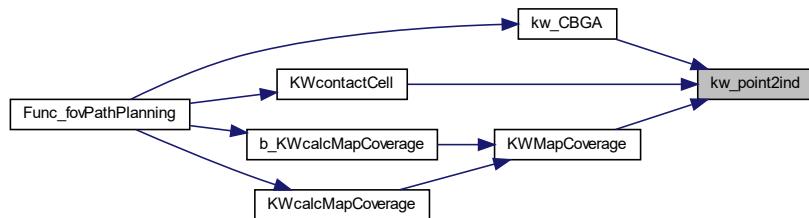
```

```
54 // Column indices
55 // delete index over
56 r.set_size(1, rIndex.size(1));
57 nx = rIndex.size(0) * rIndex.size(1);
58 for (n = 0; n < nx; n++) {
59     r[n] = (rIndex[n] < 3000.0);
60 }
61 }
62
63 r1.set_size(1, rIndex.size(1));
64 nx = rIndex.size(0) * rIndex.size(1);
65 for (n = 0; n < nx; n++) {
66     r1[n] = (rIndex[n] > 0.0);
67 }
68
69 nx = r.size(1);
70 trueCount = 0;
71 for (i = 0; i < nx; i++) {
72     if (r[i] && r1[i]) {
73         trueCount++;
74     }
75 }
76
77 r.set_size(1, cIndex.size(1));
78 nx = cIndex.size(0) * cIndex.size(1);
79 for (n = 0; n < nx; n++) {
80     r[n] = (cIndex[n] < 2000.0);
81 }
82
83 r1.set_size(1, cIndex.size(1));
84 nx = cIndex.size(0) * cIndex.size(1);
85 for (n = 0; n < nx; n++) {
86     r1[n] = (cIndex[n] > 0.0);
87 }
88
89 nx = r.size(1);
90 n = 0;
91 for (i = 0; i < nx; i++) {
92     if (r[i] && r1[i]) {
93         n++;
94     }
95 }
96
97 if (trueCount < n) {
98     n = trueCount;
99 }
100
101 if (1 > n) {
102     nx = 0;
103 } else {
104     nx = n;
105 }
106
107 rIndex.set_size(rIndex.size(0), nx);
108 if (1 > n) {
109     nx = 0;
110 } else {
111     nx = n;
112 }
113
114 cIndex.set_size(cIndex.size(0), nx);
115 if (1 > n) {
116     nx = 0;
117 } else {
118     nx = n;
119 }
120
121 b_index.set_size(1, nx);
122 for (n = 0; n < nx; n++) {
123     b_index[n] = static_cast<int>(rIndex[n]) + 3000 * (static_cast<int>(cIndex[n])
124         - 1);
125 }
126
127 // Linear indices
128 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

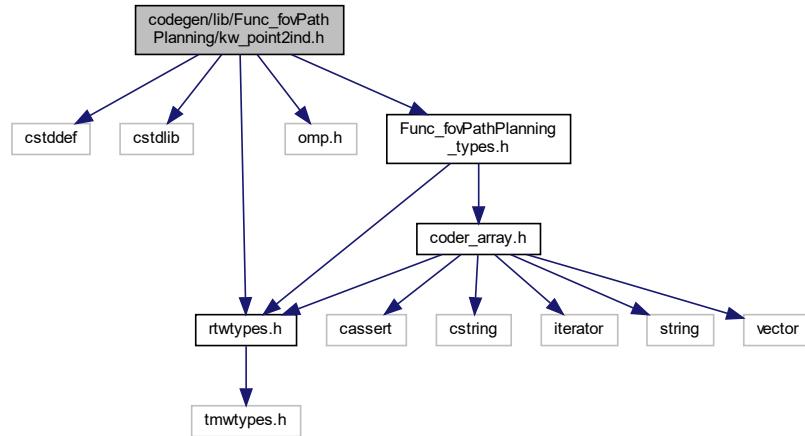


7.42 codegen/lib/Func_fovPathPlanning/kw_point2ind.h File Reference

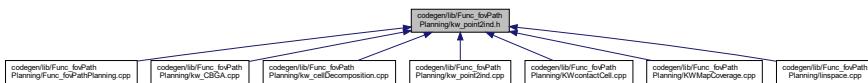
```

#include <cstdint>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
  
```

Include dependency graph for kw_point2ind.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void kw_point2ind (const double point1[2], const double point2[2], coder::array< double, 2U > &b_index)`

7.42.1 Macro Definition Documentation

7.42.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file `kw_point2ind.h`.

7.42.2 Function Documentation

7.42.2.1 kw_point2ind()

```
void kw_point2ind (
    const double point1[2],
    const double point2[2],
    coder::array< double, 2U > & b_index )
```

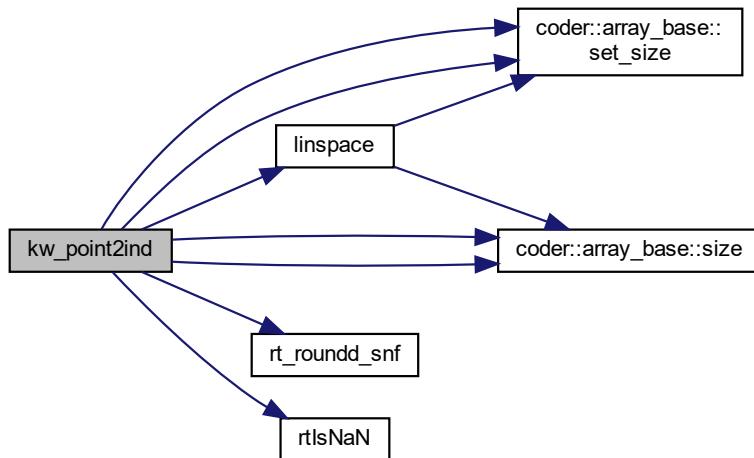
Definition at line 22 of file kw_point2ind.cpp.

```
24 {
25     double u0;
26     double u1;
27     coder::array<double, 2U> rIndex;
28     int nx;
29     int n;
30     coder::array<double, 2U> cIndex;
31     coder::array<boolean_T, 2U> r;
32     coder::array<boolean_T, 2U> rl;
33     int trueCount;
34     int i;
35     u0 = std::abs(point2[0] - point1[0]);
36     u1 = std::abs(point2[1] - point1[1]);
37     if ((!(u0 > u1)) && (!rtIsNaN(u1))) {
38         u0 = u1;
39     }
40
41     // Number of points in line
42     linspace(point1[0], point2[0], u0 + 1.0, rIndex);
43     nx = rIndex.size(1);
44     for (n = 0; n < nx; n++) {
45         rIndex[n] = rt_rounnd_snf(rIndex[n]);
46     }
47
48     // Row indices
49     linspace(point1[1], point2[1], u0 + 1.0, cIndex);
50     nx = cIndex.size(1);
51     for (n = 0; n < nx; n++) {
52         cIndex[n] = rt_rounnd_snf(cIndex[n]);
53     }
54
55     // Column indices
56     // delete index over
57     r.set_size(1, rIndex.size(1));
58     nx = rIndex.size(0) * rIndex.size(1);
59     for (n = 0; n < nx; n++) {
60         r[n] = (rIndex[n] < 3000.0);
61     }
62
63     rl.set_size(1, rIndex.size(1));
64     nx = rIndex.size(0) * rIndex.size(1);
65     for (n = 0; n < nx; n++) {
66         rl[n] = (rIndex[n] > 0.0);
67     }
68
69     nx = r.size(1);
70     trueCount = 0;
71     for (i = 0; i < nx; i++) {
72         if (r[i] && rl[i]) {
73             trueCount++;
74         }
75     }
76
77     r.set_size(1, cIndex.size(1));
78     nx = cIndex.size(0) * cIndex.size(1);
79     for (n = 0; n < nx; n++) {
80         r[n] = (cIndex[n] < 2000.0);
81     }
82
83     rl.set_size(1, cIndex.size(1));
84     nx = cIndex.size(0) * cIndex.size(1);
85     for (n = 0; n < nx; n++) {
86         rl[n] = (cIndex[n] > 0.0);
87     }
88
89     nx = r.size(1);
90     n = 0;
91     for (i = 0; i < nx; i++) {
92         if (r[i] && rl[i]) {
93             n++;
94         }
95     }
96
97     if (trueCount < n) {
98         n = trueCount;
```

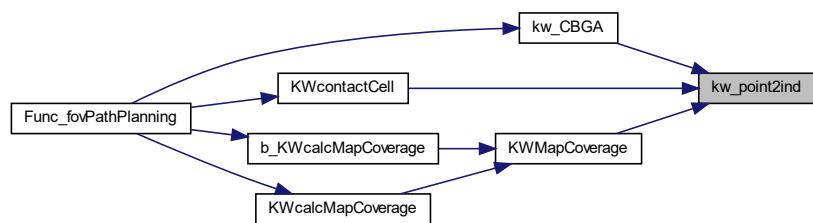
```

99      }
100     if (l > n) {
101       nx = 0;
102     } else {
103       nx = n;
104     }
105   }
106
107   rIndex.set_size(rIndex.size(0), nx);
108   if (l > n) {
109     nx = 0;
110   } else {
111     nx = n;
112   }
113
114   cIndex.set_size(cIndex.size(0), nx);
115   if (l > n) {
116     nx = 0;
117   } else {
118     nx = n;
119   }
120
121   b_index.set_size(1, nx);
122   for (n = 0; n < nx; n++) {
123     b_index[n] = static_cast<int>(rIndex[n]) + 3000 * (static_cast<int>(cIndex[n])
124     - 1);
125   }
126
127 //  Linear indices
128 }
```

Here is the call graph for this function:



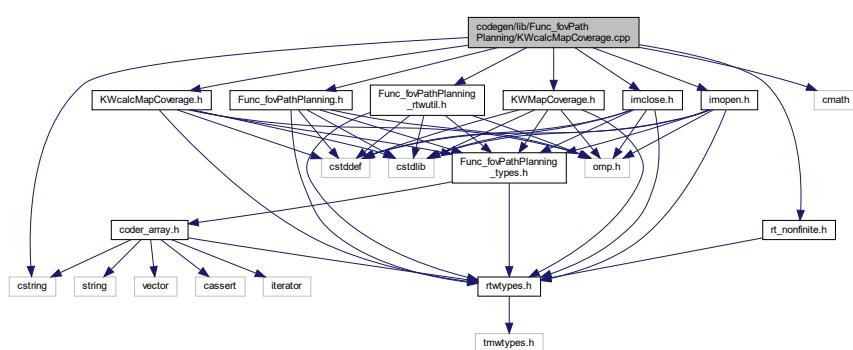
Here is the caller graph for this function:



7.43 codegen/lib(Func_fovPathPlanning/KWcalcMapCoverage.cpp) File Reference

```
#include "KWcalcMapCoverage.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "KWMMapCoverage.h"
#include "imclose.h"
#include "imopen.h"
#include "rt_nonfinite.h"
#include <cmath>
#include <cstring>
```

Include dependency graph for KWcalcMapCoverage.cpp:



Functions

- void **KWcalcMapCoverage** (const unsigned long long binaryMap[6000000], const **cell_wrap_1** coordix[3], const **cell_wrap_1** coordiy[3], unsigned long long coveredMap[6000000])
- void **b_KWcalcMapCoverage** (const unsigned long long binaryMap[6000000], **struct_T** *sensorParam, const **cell_wrap_0** coordix[3], const **cell_wrap_0** coordiy[3], unsigned long long coveredMap[6000000])

7.43.1 Function Documentation

7.43.1.1 b_KWcalcMapCoverage()

```
void b_KWcalcMapCoverage (
    const unsigned long long binaryMap[6000000],
    struct_T * sensorParam,
    const cell_wrap_0 coordix[3],
    const cell_wrap_0 coordiy[3],
    unsigned long long coveredMap[6000000] )
```

Definition at line 183 of file KWcalcMapCoverage.cpp.

```
186 {
187     static unsigned long long b_coveredMap[6000000];
188     boolean_T b[10];
```

```

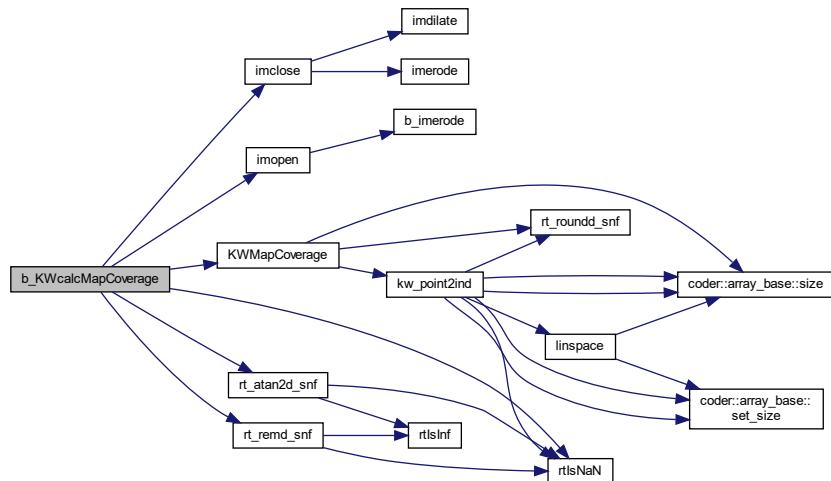
189 int k;
190 double position[2];
191 double distans_idx_0;
192 double distans_idx_1;
193 double theta;
194 double signFoV;
195 double b_position[2];
196
197 // z ft' zd' s灌(^2)
198 std::memcpy(&coveredMap[0], &binaryMap[0], 6000000U * sizeof(unsigned long
199 long));
200
201 // imshow(coveredMap);
202 // coveredMap = cat(3, binaryMap, binaryMap, binaryMap);
203 for (int robot = 0; robot < 5; robot++) {
204     int i;
205     boolean_T y;
206     boolean_T exitgl;
207     for (i = 0; i < 10; i++) {
208         b[i] = rtIsNaN(coordix[robot].f1[i]);
209     }
210
211     y = true;
212     k = 0;
213     exitgl = false;
214     while ((!exitgl) && (k < 10)) {
215         if (!b[k]) {
216             y = false;
217             exitgl = true;
218         } else {
219             k++;
220         }
221     }
222
223     if (!y) {
224         double phase;
225         position[0] = coordix[robot].f1[1];
226         position[1] = coordiy[robot].f1[1];
227         phase = 0.0;
228         for (int cont = 0; cont < 10; cont++) {
229             double delta;
230
231             // disp(['Node', num2str(cont)]);
232             distans_idx_0 = coordix[robot].f1[cont] - position[0];
233             distans_idx_1 = coordiy[robot].f1[cont] - position[1];
234             delta = std::sqrt(distans_idx_0 * distans_idx_0 + distans_idx_1 *
235             distans_idx_1) / 15.0;
236             if (delta == 0.0) {
237                 // disp([' ** KWcalcMapCoverage.m error : z$#! robot', int2str(robot)]);
238                 position[0] = coordix[robot].f1[cont];
239                 position[1] = coordiy[robot].f1[cont];
240             } else {
241                 double cphase;
242                 double d;
243
244                 // zPřcť
245                 cphase = rt_atan2d_snf(distans_idx_1, distans_idx_0);
246
247                 // yřcť
248                 theta = rt_remd_snf(cphase - phase, 6.2831853071795862);
249                 if (std::abs(theta) > 3.1415926535897931) {
250                     signFoV = theta;
251                     if (theta < 0.0) {
252                         signFoV = -1.0;
253                     } else if (theta > 0.0) {
254                         signFoV = 1.0;
255                     } else {
256                         if (theta == 0.0) {
257                             signFoV = 0.0;
258                         }
259                     }
260                     theta -= 6.2831853071795862 * signFoV;
261                 }
262
263                 signFoV = theta;
264                 if (theta < 0.0) {
265                     signFoV = -1.0;
266                 } else if (theta > 0.0) {
267                     signFoV = 1.0;
268                 } else {
269                     if (theta == 0.0) {
270                         signFoV = 0.0;
271                     }
272                 }
273             }
274             signFoV *= 1.2217304763960306;

```

```

276     d = phase + signFoV;
277     i = static_cast<int>(((phase + theta) + (signFoV - d)) / signFoV);
278     for (k = 0; k < i; k++) {
279         b_position[0] = position[0];
280         b_position[1] = position[1];
281         KWMapCoverage(sensorParam, coveredageMap, binaryMap, b_position, d +
282                         static_cast<double>(k) * signFoV);
283     }
284
285     phase = cphase;
286     distans_idx_0 /= delta;
287     distans_idx_1 /= delta;
288     i = static_cast<int>(delta);
289     for (k = 0; k < i; k++) {
290         position[0] += distans_idx_0;
291         position[1] += distans_idx_1;
292
293         // phase = phasetheta;
294         b_position[0] = position[0];
295         b_position[1] = position[1];
296         KWMapCoverage(sensorParam, coveredageMap, binaryMap, b_position,
297                         cphase);
298
299         // if flag % 1000 == 0 {
300         //     position = [ix(cont), iy(cont)];
301         //     break;
302         // end
303         // show = kw_imshow(coveredageMap, coordix, coordiy, [200, 200, 200]);
304         // title("The Robot number is " + robot,'FontSize', 22);
305     }
306 }
307 }
308 }
309 }
310
311 std::memcpy(&b_covedageMap[0], &coveredageMap[0], 6000000U * sizeof(unsigned
312 long long));
313 imclose(b_covedageMap, coveredageMap);
314 std::memcpy(&b_covedageMap[0], &coveredageMap[0], 6000000U * sizeof(unsigned
315 long long));
316 imopen(b_covedageMap, coveredageMap);
317 for (k = 0; k < 6000000; k++) {
318     if ((coveredageMap[k] < 230ULL) && (coveredageMap[k] > 0ULL)) {
319         coveredageMap[k] = 50ULL;
320     }
321 }
322 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.43.1.2 KWcalcMapCoverage()

```

void KWcalcMapCoverage (
    const unsigned long long binaryMap[6000000],
    const cell_wrap_1 coordix[3],
    const cell_wrap_1 coordiy[3],
    unsigned long long coveredageMap[6000000] )

```

Definition at line 24 of file KWcalcMapCoverage.cpp.

```

27 {
28     struct_T sensorParam;
29     static unsigned long long b_coveragedMap[6000000];
30     int loop_ub;
31     boolean_T b_data[10];
32     double position[2];
33     double distans_idx_0;
34     double distans_idx_1;
35     double theta;
36     double signFoV;
37     double b_position[2];
38     sensorParam.fov = 1.2217304763960306;
39     sensorParam.dFOV = 0.0043633231299858239;
40     sensorParam.radius = 300.0;
41     sensorParam.dvec = 15.0;
42     sensorParam.scolor = 100.0;
43     sensorParam.dc = 1.0;
44     sensorParam.Rmin = 75.0;
45     sensorParam.size[0] = 3000.0;
46     sensorParam.size[1] = 2000.0;
47     sensorParam.cs = 49.0;
48     sensorParam.mindis = 150.0;
49     sensorParam.H = 3000.0;
50     sensorParam.W = 2000.0;
51
52     // z ft' zd' s灌(^2)
53     std::memcpy(&coveredageMap[0], &binaryMap[0], 6000000U * sizeof(unsigned long
54     long));
55
56     // imshow(coveredageMap);
57     //     coveredageMap = cat(3, binaryMap, binaryMap, binaryMap);
58     for (int robot = 0; robot < 5; robot++) {
59         int b_size_idx_1;
60         int i;
61         boolean_T y;
62         boolean_T exitgl;
63         b_size_idx_1 = coordix[robot].f1.size(1);
64         loop_ub = coordix[robot].f1.size(0) * coordix[robot].f1.size(1);
65         for (i = 0; i < loop_ub; i++) {
66             b_data[i] = rtIsNaN(coordix[robot].f1[i]);
67         }
68
69         y = true;
70         loop_ub = 0;
71         exitgl = false;
72         while ((exitgl) && (loop_ub <= b_size_idx_1 - 1)) {
73             if (!b_data[loop_ub]) {
74                 y = false;
75                 exitgl = true;

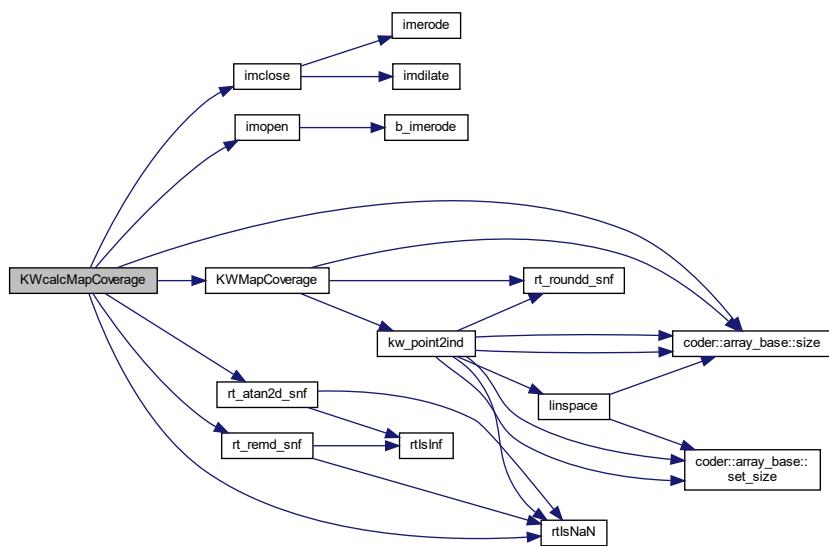
```



```

163         //  title("The Robot number is " + robot,'FontSize', 22);
164     }
165   }
166 }
167 }
168 }
169 std::memcpy(&b_coveragedMap[0], &coveredMap[0], 6000000U * sizeof(unsigned
170   long long));
171 imclose(b_coveragedMap, coveredMap);
172 std::memcpy(&b_coveragedMap[0], &coveredMap[0], 6000000U * sizeof(unsigned
173   long long));
174 imopen(b_coveragedMap, coveredMap);
175 for (loop_ub = 0; loop_ub < 6000000; loop_ub++) {
176   if ((coveredMap[loop_ub] < 230ULL) && (coveredMap[loop_ub] > 0ULL)) {
177     coveredMap[loop_ub] = 50ULL;
178   }
179 }
180 }
181 }
```

Here is the call graph for this function:



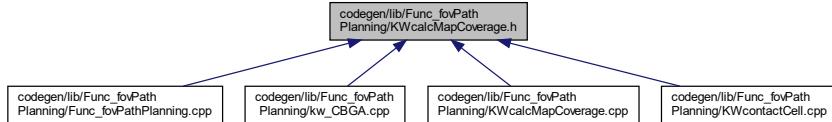
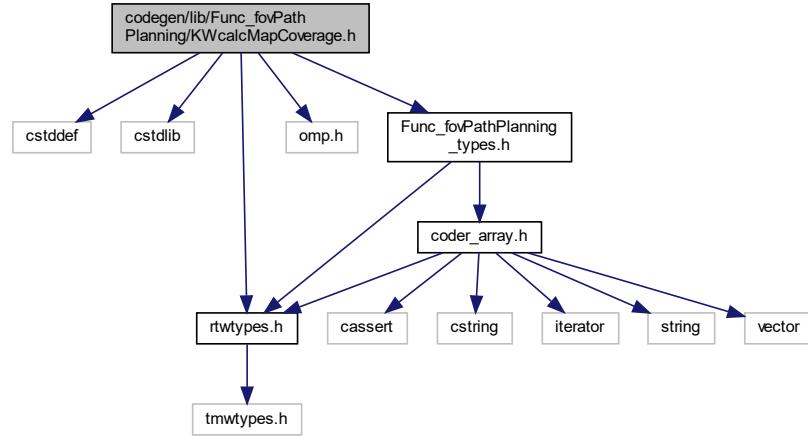
Here is the caller graph for this function:



7.44 codegen/lib/Func_fovPathPlanning/KWcalcMapCoverage.h File Reference

```
#include <cstddef>
#include <cstdlib>
```

```
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for KWcalcMapCoverage.h:
```



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void KWcalcMapCoverage (const unsigned long long binaryMap[6000000], const cell_wrap_1 coordix[3], const cell_wrap_1 coordiy[3], unsigned long long coveredMap[6000000])`
- `void b_KWcalcMapCoverage (const unsigned long long binaryMap[6000000], struct_T *sensorParam, const cell_wrap_0 coordix[3], const cell_wrap_0 coordiy[3], unsigned long long coveredMap[6000000])`

7.44.1 Macro Definition Documentation

7.44.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file KWcalcMapCoverage.h.

7.44.2 Function Documentation

7.44.2.1 b_KWcalcMapCoverage()

```
void b_KWcalcMapCoverage (
    const unsigned long long binaryMap[6000000],
    struct_T * sensorParam,
    const cell_wrap_0 coordix[3],
    const cell_wrap_0 coordiy[3],
    unsigned long long coveredMap[6000000] )
```

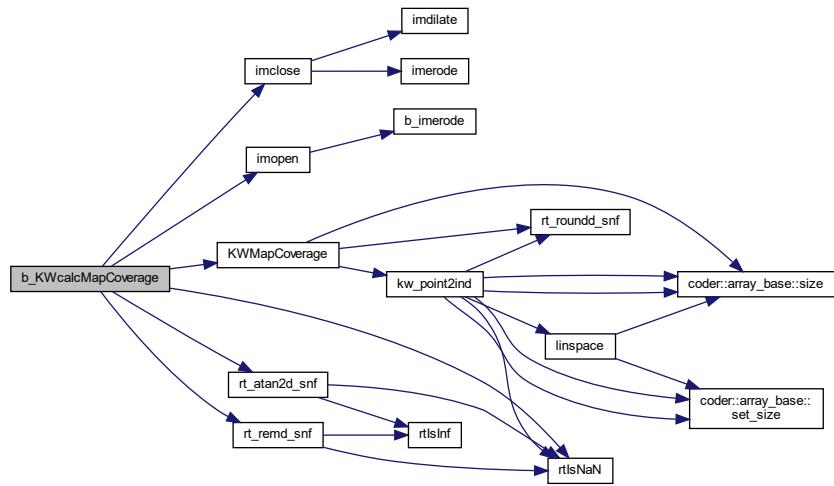
Definition at line 183 of file KWcalcMapCoverage.cpp.

```
186 {
187     static unsigned long long b_coveredMap[6000000];
188     boolean_T b[10];
189     int k;
190     double position[2];
191     double distans_idx_0;
192     double distans_idx_1;
193     double theta;
194     double signFoV;
195     double b_position[2];
196
197     // ź ft' žd' š灌(^2)
198     std::memcpy(&coveredMap[0], &binaryMap[0], 6000000U * sizeof(unsigned long
199     long));
200
201     // imshow(coveredMap);
202     //     coveredMap = cat(3, binaryMap, binaryMap, binaryMap);
203     for (int robot = 0; robot < 5; robot++) {
204         int i;
205         boolean_T y;
206         boolean_T exitg1;
207         for (i = 0; i < 10; i++) {
208             b[i] = rtIsNaN(coordix[robot].f1[i]);
209         }
210
211         y = true;
212         k = 0;
213         exitg1 = false;
214         while ((!exitg1) && (k < 10)) {
215             if (!b[k]) {
216                 y = false;
217                 exitg1 = true;
218             } else {
219                 k++;
220             }
221         }
222
223         if (!y) {
224             double phase;
225             position[0] = coordix[robot].f1[1];
226             position[1] = coordiy[robot].f1[1];
227             phase = 0.0;
228             for (int cont = 0; cont < 10; cont++) {
229                 double delta;
230
231                 // disp(['Node', num2str(cont)]);
232                 distans_idx_0 = coordix[robot].f1[cont] - position[0];
233                 distans_idx_1 = coordiy[robot].f1[cont] - position[1];
234                 delta = std::sqrt(distans_idx_0 * distans_idx_0 + distans_idx_1 *
235                                 distans_idx_1) / 15.0;
236                 if (delta == 0.0) {
```

```

237     // disp([' ** KWcalcMapCoverage.m error : zšlo ž!! robot', int2str(robot)]);
238     position[0] = coordix[robot].f1[cont];
239     position[1] = coordiy[robot].f1[cont];
240 } else {
241     double cphase;
242     double d;
243
244     // Přečít
245     cphase = rt_atan2d_snf(distans_idx_1, distans_idx_0);
246
247     // řečt
248     theta = rt_remd_snf(cphase - phase, 6.2831853071795862);
249     if (std::abs(theta) > 3.1415926535897931) {
250         signFoV = theta;
251         if (theta < 0.0) {
252             signFoV = -1.0;
253         } else if (theta > 0.0) {
254             signFoV = 1.0;
255         } else {
256             if (theta == 0.0) {
257                 signFoV = 0.0;
258             }
259         }
260
261         theta -= 6.2831853071795862 * signFoV;
262     }
263
264     signFoV = theta;
265     if (theta < 0.0) {
266         signFoV = -1.0;
267     } else if (theta > 0.0) {
268         signFoV = 1.0;
269     } else {
270         if (theta == 0.0) {
271             signFoV = 0.0;
272         }
273     }
274
275     signFoV *= 1.2217304763960306;
276     d = phase + signFoV;
277     i = static_cast<int>((phase + theta) + (signFoV - d)) / signFoV;
278     for (k = 0; k < i; k++) {
279         b_position[0] = position[0];
280         b_position[1] = position[1];
281         KWMapCoverage(sensorParam, coveredMap, binaryMap, b_position, d +
282                         static_cast<double>(k) * signFoV);
283     }
284
285     phase = cphase;
286     distans_idx_0 /= delta;
287     distans_idx_1 /= delta;
288     i = static_cast<int>(delta);
289     for (k = 0; k < i; k++) {
290         position[0] += distans_idx_0;
291         position[1] += distans_idx_1;
292
293         // phase = phase+theta;
294         b_position[0] = position[0];
295         b_position[1] = position[1];
296         KWMapCoverage(sensorParam, coveredMap, binaryMap, b_position,
297                         cphase);
298
299         // if flag % úkff želá t' když; ggúijžčť it'
300         // position = [ix(cont), iy(cont)];
301         // break;
302         // end
303         // show = kw_imshow(coveredMap, coordix, coordiy, [200, 200, 200]);
304         // title("The Robot number is " + robot,'FontSize', 22);
305     }
306 }
307 }
308 }
309 }
310
311 std::memcpy(&b_coveredMap[0], &coveredMap[0], 6000000U * sizeof(unsigned
312 long long));
313 imclose(b_coveredMap, coveredMap);
314 std::memcpy(&b_coveredMap[0], &coveredMap[0], 6000000U * sizeof(unsigned
315 long long));
316 imopen(b_coveredMap, coveredMap);
317 for (k = 0; k < 6000000; k++) {
318     if ((coveredMap[k] < 230ULL) && (coveredMap[k] > 0ULL)) {
319         coveredMap[k] = 50ULL;
320     }
321 }
322 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.44.2.2 KWcalcMapCoverage()

```
void KWcalcMapCoverage (
    const unsigned long long binaryMap[6000000],
    const cell_wrap_1 coordix[3],
    const cell_wrap_1 coordiy[3],
    unsigned long long coveredMap[6000000] )
```

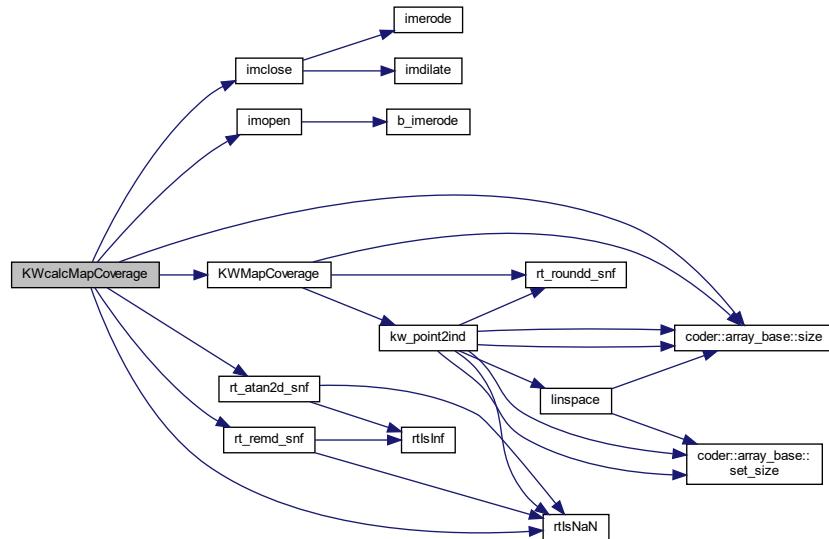
Definition at line 24 of file KWcalcMapCoverage.cpp.

```
27 {
28     struct_T sensorParam;
29     static unsigned long long b_covaragedMap[6000000];
30     int loop_ub;
31     boolean_T b_data[10];
32     double position[2];
33     double distans_idx_0;
34     double distans_idx_1;
35     double theta;
36     double signFoV;
37     double b_position[2];
38     sensorParam.fov = 1.2217304763960306;
39     sensorParam.dFOV = 0.0043633231299858239;
40     sensorParam.radius = 300.0;
```



```
128         } else {
129             if (theta == 0.0) {
130                 signFoV = 0.0;
131             }
132         }
133
134         signFoV *= 1.2217304763960306;
135         d = phase + signFoV;
136         loop_ub = static_cast<int>(((phase + theta) + (signFoV - d)) / signFoV);
137         for (b_size_idx_1 = 0; b_size_idx_1 < loop_ub; b_size_idx_1++) {
138             b_position[0] = position[0];
139             b_position[1] = position[1];
140             KWMapCoverage(&sensorParam, coveredMap, binaryMap, b_position, d +
141                           static_cast<double>(b_size_idx_1) * signFoV);
142         }
143
144         phase = cphase;
145         distans_idx_0 /= delta;
146         distans_idx_1 /= delta;
147         loop_ub = static_cast<int>(delta);
148         for (b_size_idx_1 = 0; b_size_idx_1 < loop_ub; b_size_idx_1++) {
149             position[0] += distans_idx_0;
150             position[1] += distans_idx_1;
151
152             // phase = phasetheta;
153             b_position[0] = position[0];
154             b_position[1] = position[1];
155             KWMapCoverage(&sensorParam, coveredMap, binaryMap, b_position,
156                           cphase);
157
158             // if flag % 1000 == 0 {
159             //     position = [ix(cont), iy(cont)];
160             //     break;
161             // end
162             // show = kw_imshow(coveredMap, coordix, coordiy, [200, 200, 200]);
163             // title("The Robot number is " + robot,'FontSize', 22);
164         }
165     }
166 }
167 }
168 }
169
170 std::memcpy(&b_coveredMap[0], &coveredMap[0], 6000000U * sizeof(unsigned
171 long long));
172 imclose(b_coveredMap, coveredMap);
173 std::memcpy(&b_coveredMap[0], &coveredMap[0], 6000000U * sizeof(unsigned
174 long long));
175 imopen(b_coveredMap, coveredMap);
176 for (loop_ub = 0; loop_ub < 6000000; loop_ub++) {
177     if ((coveredMap[loop_ub] < 230ULL) && (coveredMap[loop_ub] > 0ULL)) {
178         coveredMap[loop_ub] = 50ULL;
179     }
180 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

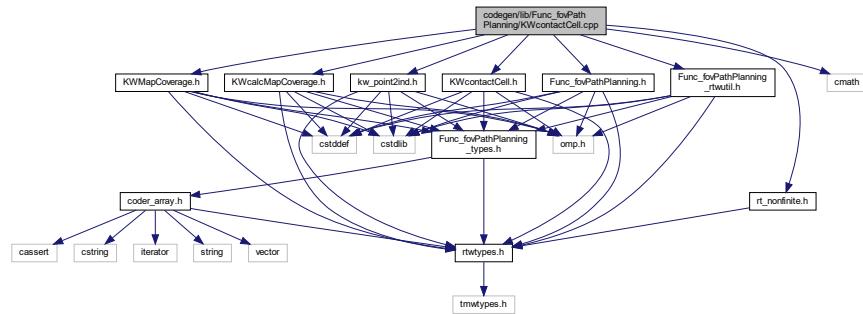


7.45 codegen/lib/Func_fovPathPlanning/KWcontactCell.cpp File Reference

```

#include "KWcontactCell.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "KWMpCoverage.h"
#include "KWcalcMapCoverage.h"
#include "kw_point2ind.h"
#include "rt_nonfinite.h"
#include <cmath>
  
```

Include dependency graph for KWcontactCell.cpp:



Functions

- void **KWcontactCell** (const unsigned long long binaryMap[6000000], const **cell_wrap_1** cluster[3], double num, const **cell_wrap_1** coordix[3], const **cell_wrap_1** coordiy[3], double *cll, double *robot, double *node, double point[2], double *phase)

7.45.1 Function Documentation

7.45.1.1 KWcontactCell()

```
void KWcontactCell (
    const unsigned long long binaryMap[6000000],
    const cell_wrap_1 cluster[3],
    double num,
    const cell_wrap_1 coordix[3],
    const cell_wrap_1 coordiy[3],
    double * cl,
    double * robot,
    double * node,
    double point[2],
    double * phase )
```

Definition at line 23 of file KWcontactCell.cpp.

```

27 {
28     int celln;
29     double b_min;
30     double position_fnode[2];
31     double position_cell[2];
32     double theta;
33     coder::array<double, 2U> b_index;
34
35     // cllž cluster ž
36     *cll = 0.0;
37     *robot = 0.0;
38     *node = 0.0;
39     point[0] = 0.0;
40     point[1] = 0.0;
41     *phase = 0.0;
42     celln = -1;
43     b_min = 5000.0;
44     for (int robotn = 0; robotn < 5; robotn++) {
45         int phase_tmp;

```

```

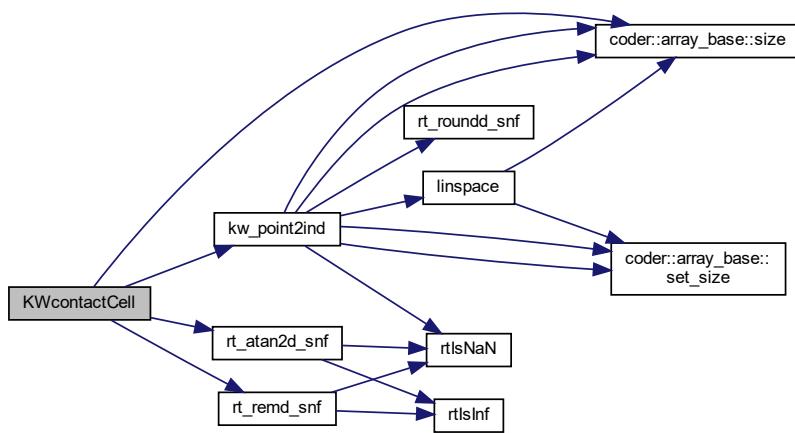
46     phase_tmp = coordix[robotn].f1.size(1);
47     for (int cont = 0; cont <= phase_tmp - 2; cont++) {
48         if (cont + 2 != coordix[robotn].f1.size(1)) {
49             double position_inode_idx_0;
50             double position_inode_idx_1;
51             int i;
52             position_inode_idx_0 = coordix[robotn].f1[cont];
53             position_inode_idx_1 = coordiy[robotn].f1[cont];
54             position_fnode[0] = coordix[robotn].f1[cont + 1];
55             position_fnode[1] = coordiy[robotn].f1[cont + 1];
56             i = static_cast<int>(num);
57             for (int k = 0; k < i; k++) {
58                 int il;
59                 il = static_cast<int>((static_cast<double>(cluster[k].f1.size(0) *
60                     cluster[k].f1.size(1)) + 1.0) / 2.0);
61                 for (int cn = 0; cn < il; cn++) {
62                     int b_cn;
63                     double distans;
64                     double zl_idx_0;
65                     b_cn = cn << 1;
66                     position_cell[0] = cluster[k].f1[b_cn];
67                     position_cell[1] = cluster[k].f1[b_cn + 1];
68
69                     // ýřčt
70                     // ýřčt
71                     theta = rt_remd_snf(rt_atan2d_snf(position_fnode[1], position_fnode
72                         [0]) - rt_atan2d_snf(position_inode_idx_1, position_inode_idx_0),
73                         6.2831853071795862);
74                     if (std::abs(theta) > 3.1415926535897931) {
75                         distans = theta;
76                         if (theta < 0.0) {
77                             distans = -1.0;
78                         } else if (theta > 0.0) {
79                             distans = 1.0;
80                         } else {
81                             if (theta == 0.0) {
82                                 distans = 0.0;
83                             }
84                         }
85
86                         theta -= 6.2831853071795862 * distans;
87                     }
88
89                     distans = position_fnode[0] - cluster[k].f1[b_cn];
90                     zl_idx_0 = distans * distans;
91                     distans = position_fnode[1] - cluster[k].f1[b_cn + 1];
92                     distans = std::sqrt(zl_idx_0 + distans * distans) + std::abs(theta -
93                         30.0) * 5.0;
94                     if ((b_min > distans) && (distans > 150.0)) {
95                         int flag;
96                         int s;
97                         boolean_T exitgl;
98                         kw_point2ind(position_fnode, position_cell, b_index);
99                         flag = 1;
100                        s = 0;
101                        exitgl = false;
102                        while (!exitgl) && (s <= static_cast<int>(b_index[0]) - 1) {
103                            if (binaryMap[s] < 100ULL) {
104                                // 付žří ýyšš
105                                flag = 0;
106                                exitgl = true;
107                            } else {
108                                s++;
109                            }
110
111                            if (flag != 0) {
112                                b_min = distans;
113                                *cll = static_cast<double>(k) + 1.0;
114
115                                // cluster ž
116                                celln = b_cn;
117
118                                // cell ž
119                                *robot = static_cast<double>(robotn) + 1.0;
120                                *node = static_cast<double>(cont) + 2.0;
121                                point[0] = position_fnode[0];
122                                point[1] = position_fnode[1];
123
124                                // ž ž ž
125                                // delta = [cell(1) + cell(3), cell(2) + cell(4)]/2-position_node;
126                                // » fžt'
127                                // » fžt'
128                            }
129                        }
130                    }
131    }
132}

```

```

133     }
134
135     if (*c11 != 0.0) {
136         phase_tmp = static_cast<int>(*c11) - 1;
137         *phase = rt_atan2d_snf(cluster[phase_tmp].f1[celln + 1] - point[1],
138             cluster[phase_tmp].f1[celln] - point[0]);
139
140         // Cells(Cells==nearCell(1),:) = [];
141         // řešení celíř node ;ž ů
142     }
143 }
144 }
```

Here is the call graph for this function:



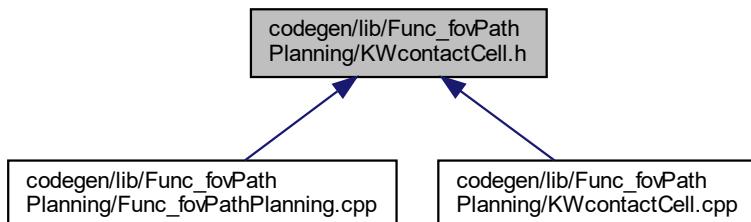
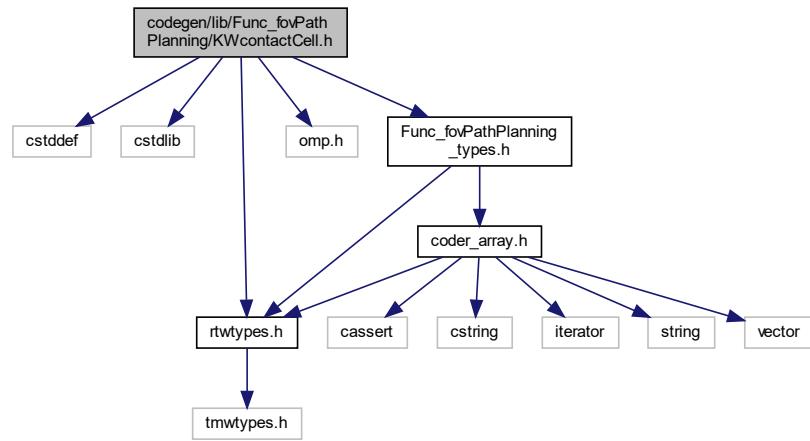
Here is the caller graph for this function:



7.46 codegen/lib/Func_fovPathPlanning/KWcontactCell.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
```

```
#include "Func_fovPathPlanning_types.h"
Include dependency graph for KWcontactCell.h:
```



Macros

- `#define MAX_THREADS omp_get_max_threads()`

Functions

- `void KWcontactCell (const unsigned long long binaryMap[6000000], const cell_wrap_1 cluster[3], double num, const cell_wrap_1 coordix[3], const cell_wrap_1 coordiy[3], double *cll, double *robot, double *node, double point[2], double *phase)`

7.46.1 Macro Definition Documentation

7.46.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file KWcontactCell.h.

7.46.2 Function Documentation

7.46.2.1 KWcontactCell()

```
void KWcontactCell (
    const unsigned long long binaryMap[6000000],
    const cell_wrap_1 cluster[3],
    double num,
    const cell_wrap_1 coordix[3],
    const cell_wrap_1 coordiy[3],
    double * cll,
    double * robot,
    double * node,
    double point[2],
    double * phase )
```

Definition at line 23 of file KWcontactCell.cpp.

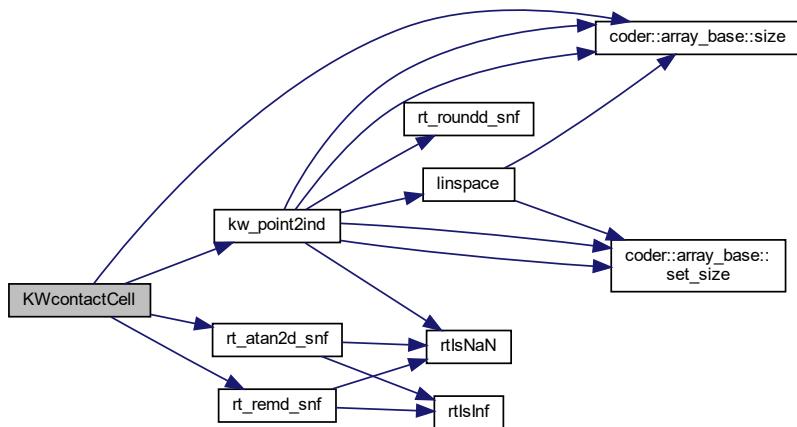
```
27 {
28     int celln;
29     double b_min;
30     double position_fnode[2];
31     double position_cell[2];
32     double theta;
33     coder::array<double, 2U> b_index;
34
35     // cllž cluster ź
36     *cll = 0.0;
37     *robot = 0.0;
38     *node = 0.0;
39     point[0] = 0.0;
40     point[1] = 0.0;
41     *phase = 0.0;
42     celln = -1;
43     b_min = 5000.0;
44     for (int robotn = 0; robotn < 5; robotn++) {
45         int phase_tmp;
46         phase_tmp = coordix[robotn].f1.size(1);
47         for (int cont = 0; cont <= phase_tmp - 2; cont++) {
48             if (cont + 2 != coordix[robotn].f1.size(1)) {
49                 double position_inode_idx_0;
50                 double position_inode_idx_1;
51                 int i;
52                 position_inode_idx_0 = coordix[robotn].f1[cont];
53                 position_inode_idx_1 = coordiy[robotn].f1[cont];
54                 position_fnode[0] = coordix[robotn].f1[cont + 1];
55                 position_fnode[1] = coordiy[robotn].f1[cont + 1];
56                 i = static_cast<int>(num);
57                 for (int k = 0; k < i; k++) {
58                     int il;
59                     il = static_cast<int>((static_cast<double>(cluster[k].f1.size(0) *
60                                         cluster[k].f1.size(1)) + 1.0) / 2.0);
61                     for (int cn = 0; cn < il; cn++) {
62                         int b_cn;
63                         double distans;
64                         double z1_idx_0;
65                         b_cn = cn << 1;
66                         position_cell[0] = cluster[k].f1[b_cn];
67                         position_cell[1] = cluster[k].f1[b_cn + 1];
68
69                         // ýřečt
70                         // ýřečt
```

```

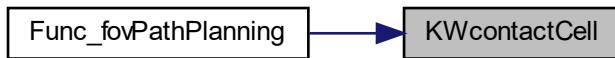
1 theta = rt_remnd_snf(rt_atan2d_snf(position_fnode[1], position_fnode
2 [0]) - rt_atan2d_snf(position_inode_idx_1, position_inode_idx_0),
3 6.2831853071795862);
4 if (std::abs(theta) > 3.1415926535897931) {
5     distans = theta;
6     if (theta < 0.0) {
7         distans = -1.0;
8     } else if (theta > 0.0) {
9         distans = 1.0;
10    } else {
11        if (theta == 0.0) {
12            distans = 0.0;
13        }
14    }
15
16    theta -= 6.2831853071795862 * distans;
17}
18
19 distans = position_fnode[0] - cluster[k].f1[b_cn];
20 z1_idx_0 = distans * distans;
21 distans = position_fnode[1] - cluster[k].f1[b_cn + 1];
22 distans = std::sqrt(z1_idx_0 + distans * distans) + std::abs(theta -
23 30.0) * 5.0;
24 if ((b_min > distans) && (distans > 150.0)) {
25     int flag;
26     int s;
27     boolean_T exitg1;
28     kw_point2ind(position_fnode, position_cell, b_index);
29     flag = 1;
30     s = 0;
31     exitg1 = false;
32     while ((!exitg1) && (s <= static_cast<int>(b_index[0]) - 1)) {
33         if (binaryMap[s] < 100ULL) {
34             // 付箋ř ýÿšš
35             flag = 0;
36             exitg1 = true;
37         } else {
38             s++;
39         }
40
41         if (flag != 0) {
42             b_min = distans;
43             *c1l = static_cast<double>(k) + 1.0;
44
45             //  cluster ź
46             celln = b_cn;
47
48             //  cell ź
49             *robot = static_cast<double>(robotn) + 1.0;
50             *node = static_cast<double>(cont) + 2.0;
51             point[0] = position_fnode[0];
52             point[1] = position_fnode[1];
53
54             //  ź ź ź
55             // delta = [cell(1) + cell(3), cell(2) + cell(4)]/2-position_node;
56             //  » f呈t'
57         }
58     }
59 }
60
61 }
62
63 }
64
65 if (*c1l != 0.0) {
66     phase_tmp = static_cast<int>(*c1l) - 1;
67     *phase = rt_atan2d_snf(cluster[phase_tmp].f1[celln + 1] - point[1],
68     cluster[phase_tmp].f1[celln] - point[0]);
69
70     //      Cells(Cells==nearCell(1),:) = [];
71     //  řaaš cellř node ;ž ū
72 }
73
74 }
75
76 }
77
78 }
79
80 }
81
82 }
83
84 }
85
86 }
87
88 }
89
90 }
91
92 }
93
94 }
95
96 }
97
98 }
99
100 }
101
102 }
103
104 }
105
106 }
107
108 }
109
110 }
111
112 }
113
114 }
115
116 }
117
118 }
119
120 }
121
122 }
123
124
125 }
126
127 }
128
129 }
130
131 }
132
133 }
134
135 if (*c1l != 0.0) {
136     phase_tmp = static_cast<int>(*c1l) - 1;
137     *phase = rt_atan2d_snf(cluster[phase_tmp].f1[celln + 1] - point[1],
138     cluster[phase_tmp].f1[celln] - point[0]);
139
140     //      Cells(Cells==nearCell(1),:) = [];
141     //  řaaš cellř node ;ž ū
142 }
143
144 }

```

Here is the call graph for this function:



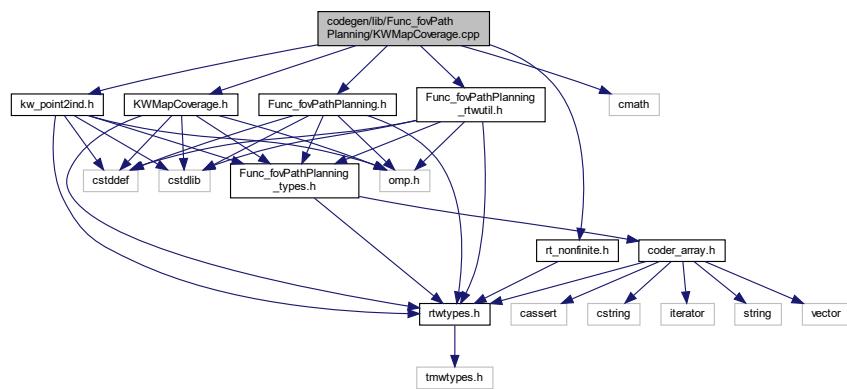
Here is the caller graph for this function:



7.47 codegen/lib/Func_fovPathPlanning/KWMapCoverage.cpp File Reference

```
#include "KWMapCoverage.h"
#include "Func_fovPathPlanning.h"
#include "Func_fovPathPlanning_rtwutil.h"
#include "kw_point2ind.h"
#include "rt_nonfinite.h"
#include <cmath>
```

Include dependency graph for KWMapCoverage.cpp:



Functions

- double **KWMapCoverage** (struct_T *sensorParam, unsigned long long coveredMap[6000000], const unsigned long long binaryMap[6000000], double position[2], double phase)

7.47.1 Function Documentation

7.47.1.1 KWMapCoverage()

```
double KWMapCoverage (
    struct_T * sensorParam,
    unsigned long long coveredMap[6000000],
    const unsigned long long binaryMap[6000000],
    double position[2],
    double phase )
```

Definition at line 21 of file KWMapCoverage.cpp.

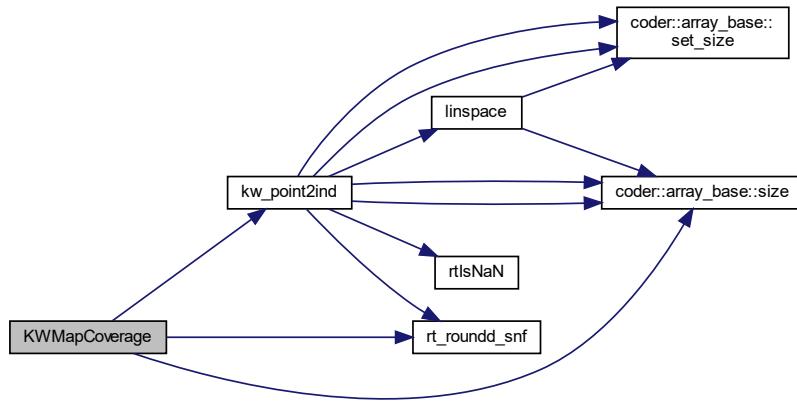
```

24 {
25     double flag;
26     double b_position[2];
27     coder::array<double, 2U> b_index;
28     position[0] = rt_roundd_snf(position[0]);
29     position[1] = rt_roundd_snf(position[1]);
30     sensorParam->scolor += sensorParam->dc;
31     sensorParam->dc += static_cast<double>(((sensorParam->scolor < 51.0) +
32         (sensorParam->scolor < 51.0)) - (sensorParam->scolor > 169.0)) -
33         (sensorParam->scolor > 169.0));
34     if (binaryMap[(static_cast<int>(position[0]) + 3000 * (static_cast<int>
35         (position[1]) - 1)) - 1] == 0ULL) {
36         // disp('error : žáž ū řá t' KW_MapCoverage.m');
37         flag = 1.0;
38     } else {
39         int i;
40         i = static_cast<int>(((phase + 0.6108652381980153) + (0.0043633231299858239
41             - (phase - 0.6108652381980153))) / 0.0043633231299858239);
42         for (int theta = 0; theta < i; theta++) {
43             double b_theta;
44             int il;
45             int s;
46             boolean_T exitgl;
47             b_theta = (phase - 0.6108652381980153) + static_cast<double>(theta) *
```

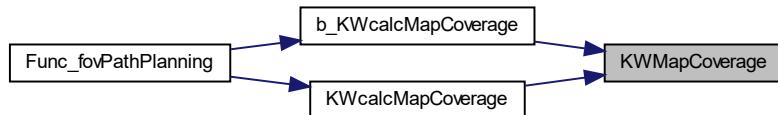
```

48     0.0043633231299858239;
49     b_position[0] = position[0] + 300.0 * std::cos(b_theta);
50     b_position[1] = position[1] + 300.0 * std::sin(b_theta);
51     kw_point2ind(position, b_position, b_index);
52
53     // Linear indices
54     il = static_cast<int>(b_index[b_index.size(1) - 1] + (1.0 - b_index[0]));
55     s = 0;
56     exitgl = false;
57     while ((!exitgl) && (s <= il - 1)) {
58         int i2;
59         i2 = static_cast<int>(b_index[0] + static_cast<double>(s) - 1);
60         if (binaryMap[i2] < 100ULL) {
61             // Тјзир јијш
62             exitgl = true;
63         } else {
64             unsigned long long u;
65             b_theta = rt_roundd_snf(sensorParam->scolor);
66             if (b_theta < 1.8446744073709552E+19) {
67                 if (b_theta >= 0.0) {
68                     u = static_cast<unsigned long long>(b_theta);
69                 } else {
70                     u = 0ULL;
71                 }
72             } else if (b_theta >= 1.8446744073709552E+19) {
73                 u = MAX_uint64_T;
74             } else {
75                 u = 0ULL;
76             }
77             coveredageMap[i2] = u;
78             s++;
79         }
80     }
81 }
82
83 flag = 0.0;
84 }
85
86 return flag;
87 }
88 }
```

Here is the call graph for this function:

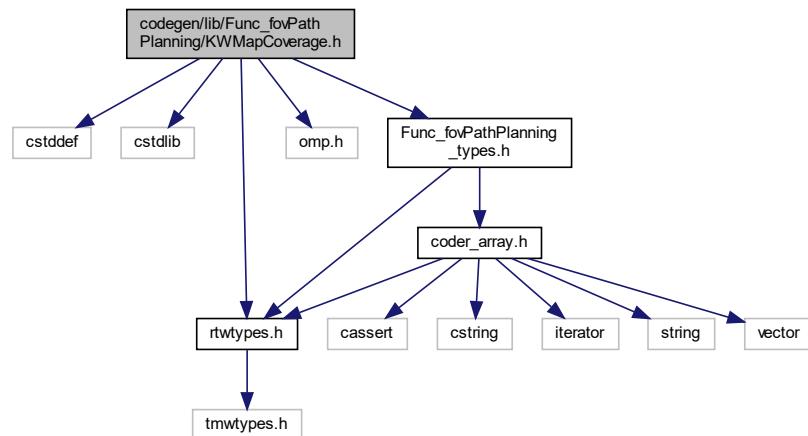


Here is the caller graph for this function:



7.48 codegen/lib/Func_fovPathPlanning/KWMapCoverage.h File Reference

```
#include <cstddef>
#include <cstdlib>
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for KWMapCoverage.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define MAX_THREADS omp_get_max_threads()

Functions

- double KWMapCoverage (struct_T *sensorParam, unsigned long long coveredMap[6000000], const unsigned long long binaryMap[6000000], double position[2], double phase)

7.48.1 Macro Definition Documentation

7.48.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file KWMapCoverage.h.

7.48.2 Function Documentation

7.48.2.1 KWMapCoverage()

```
double KWMapCoverage (
    struct_T * sensorParam,
    unsigned long long coveredMap[6000000],
    const unsigned long long binaryMap[6000000],
    double position[2],
    double phase )
```

Definition at line 21 of file KWMapCoverage.cpp.

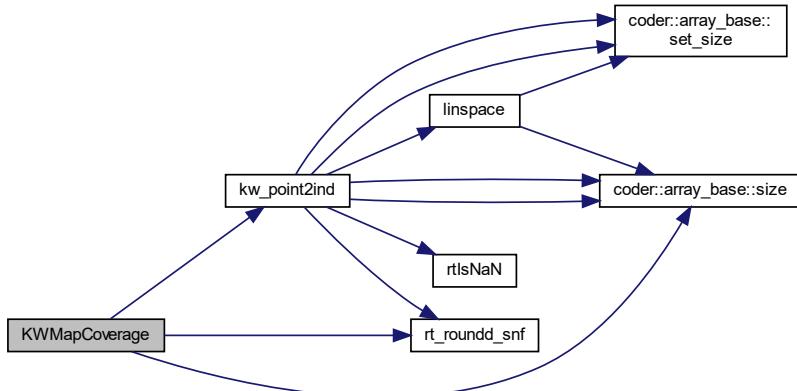
```
24 {
25     double flag;
26     double b_position[2];
27     coder::array<double, 2U> b_index;
28     position[0] = rt_roundd_snf(position[0]);
29     position[1] = rt_roundd_snf(position[1]);
30     sensorParam->scolor += sensorParam->dc;
31     sensorParam->dc += static_cast<double>(((sensorParam->scolor < 51.0) +
32         (sensorParam->scolor < 51.0)) - (sensorParam->scolor > 169.0)) -
33         (sensorParam->scolor > 169.0));
34     if (binaryMap[(static_cast<int>(position[0]) + 3000 * (static_cast<int>
35         (position[1]) - 1)) - 1] == OULL) {
36         // disp('error : žóž ūřířa t' KW_MapCoverage.m');
37         flag = 1.0;
38     } else {
39         int i;
40         i = static_cast<int>((phase + 0.6108652381980153) + (0.0043633231299858239 *
41             -(phase - 0.6108652381980153))) / 0.0043633231299858239;
42         for (int theta = 0; theta < i; theta++) {
43             double b_theta;
44             int il;
45             int s;
46             boolean_T exitgl;
47             b_theta = (phase - 0.6108652381980153) + static_cast<double>(theta) *
48                 0.0043633231299858239;
49             b_position[0] = position[0] + 300.0 * std::cos(b_theta);
50             b_position[1] = position[1] + 300.0 * std::sin(b_theta);
51             kw_point2ind(position, b_position, b_index);
52
53             // Linear indices
54             il = static_cast<int>(b_index[b_index.size(1) - 1] + (1.0 - b_index[0]));
55             s = 0;
56             exitgl = false;
```

```

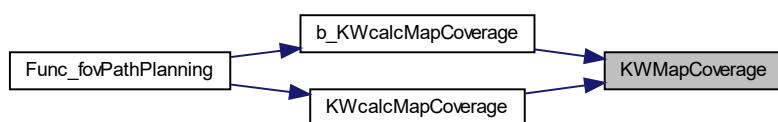
57     while ((!exitg1) && (s <= il - 1)) {
58         int i2;
59         i2 = static_cast<int>(b_index[0] + static_cast<double>(s)) - 1;
60         if (binaryMap[i2] < 100ULL) {
61             // 将 z 到 y 转换为
62             exitg1 = true;
63         } else {
64             unsigned long long u;
65             b_theta = rt_roundd_snf(sensorParam->scolor);
66             if (b_theta < 1.8446744073709552E+19) {
67                 if (b_theta >= 0.0) {
68                     u = static_cast<unsigned long long>(b_theta);
69                 } else {
70                     u = 0ULL;
71                 }
72             } else if (b_theta >= 1.8446744073709552E+19) {
73                 u = MAX_uint64_T;
74             } else {
75                 u = 0ULL;
76             }
77             coveredMap[i2] = u;
78             s++;
79         }
80     }
81 }
82
83 flag = 0.0;
84
85 }
86
87 return flag;
88 }

```

Here is the call graph for this function:

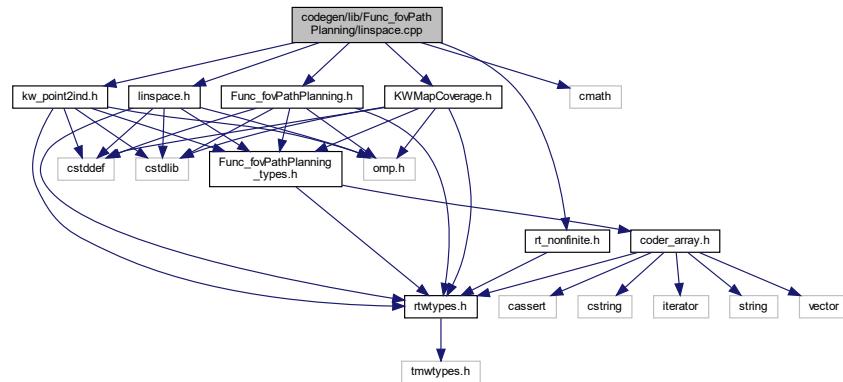


Here is the caller graph for this function:



7.49 codegen/lib/Func_fovPathPlanning/linspace.cpp File Reference

```
#include "linspace.h"
#include "Func_fovPathPlanning.h"
#include "KWMapCoverage.h"
#include "kw_point2ind.h"
#include "rt_nonfinite.h"
#include <cmath>
Include dependency graph for linspace.cpp:
```



Functions

- void [linspace](#) (double d1, double d2, double n1, [coder::array< double, 2U > &y\)](#)

7.49.1 Function Documentation

7.49.1.1 linspace()

```
void linspace (
    double d1,
    double d2,
    double n1,
    coder::array< double, 2U > & y \)
```

Definition at line 21 of file [linspace.cpp](#).

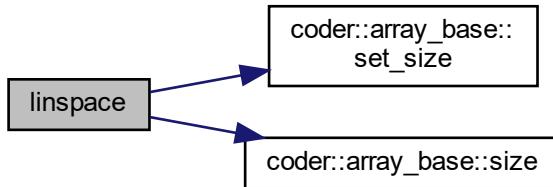
```
22 {
23     int i;
24     int y_tmp;
25     i = static\_cast<int>(std::floor(n1));
26     y.set\_size(1, i);
27     y_tmp = i - 1;
28     y[y_tmp] = d2;
29     if (y.size(1) >= 2) {
30         y[0] = d1;
31         if (y.size(1) >= 3) {
32             if ((d1 == -d2) && (i > 2)) {
33                 for (int k = 2; k <= y_tmp; k++) {
34                     y[k - 1] = d2 * (static\_cast<double>((k << 1) - i) - 1) / (
```

```

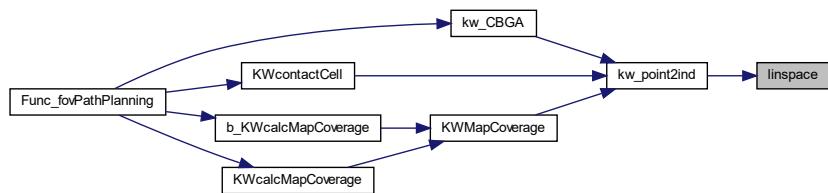
35         static_cast<double>(i) - 1.0));
36     }
37
38     if ((i & 1) == 1) {
39         y[i >> 1] = 0.0;
40     }
41 } else if (((d1 < 0.0) != (d2 < 0.0)) && ((std::abs(d1) >
42             8.9884656743115785E+307) || (std::abs(d2) >
43             8.9884656743115785E+307))) {
44     double delta1;
45     double delta2;
46     delta1 = d1 / (static_cast<double>(y.size(1)) - 1.0);
47     delta2 = d2 / (static_cast<double>(y.size(1)) - 1.0);
48     i = y.size(1);
49     for (int k = 0; k <= i - 3; k++) {
50         y[k + 1] = (d1 + delta2 * (static_cast<double>(k + 1.0)) - delta1 * (
51             static_cast<double>(k) + 1.0));
52     }
53 } else {
54     double delta1;
55     delta1 = (d2 - d1) / (static_cast<double>(y.size(1)) - 1.0);
56     i = y.size(1);
57     for (int k = 0; k <= i - 3; k++) {
58         y[k + 1] = d1 + (static_cast<double>(k + 1.0) * delta1);
59     }
60 }
61 }
62 }
63 }

```

Here is the call graph for this function:



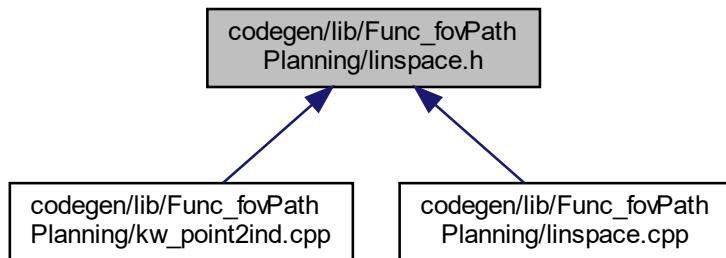
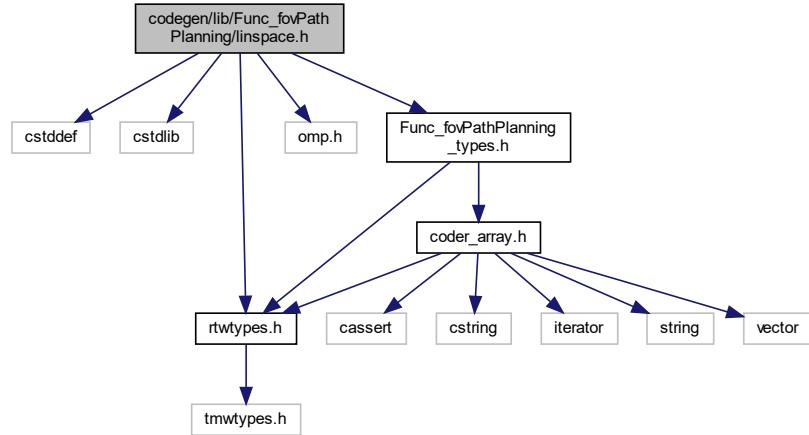
Here is the caller graph for this function:



7.50 codegen/lib/Func_fovPathPlanning/linspace.h File Reference

```
#include <cstdint>
#include <cstdlib>
```

```
#include "rtwtypes.h"
#include "omp.h"
#include "Func_fovPathPlanning_types.h"
Include dependency graph for linspace.h:
```



Macros

- #define MAX_THREADS omp_get_max_threads()

Functions

- void **linspace** (double d1, double d2, double n1, **coder::array< double, 2U > &y**)

7.50.1 Macro Definition Documentation

7.50.1.1 MAX_THREADS

```
#define MAX_THREADS omp_get_max_threads()
```

Definition at line 21 of file linspace.h.

7.50.2 Function Documentation

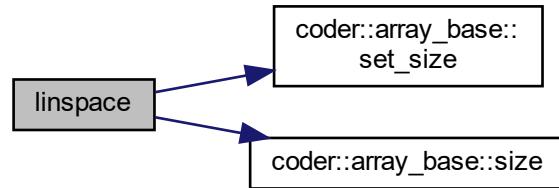
7.50.2.1 linspace()

```
void linspace (
    double d1,
    double d2,
    double n1,
    coder::array< double, 2U > & y )
```

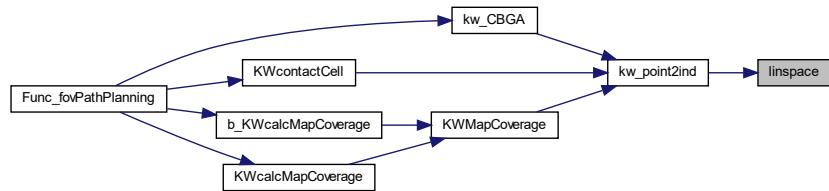
Definition at line 21 of file linspace.cpp.

```
22 {
23     int i;
24     int y_tmp;
25     i = static_cast<int>(std::floor(n1));
26     y.set_size(1, i);
27     y_tmp = i - 1;
28     y[y_tmp] = d2;
29     if (y.size(1) >= 2) {
30         y[0] = d1;
31         if (y.size(1) >= 3) {
32             if ((d1 == -d2) && (i > 2)) {
33                 for (int k = 2; k <= y_tmp; k++) {
34                     y[k - 1] = d2 * (static_cast<double>(((k << 1) - i) - 1) /
35                         static_cast<double>(i) - 1.0));
36             }
37             if ((i & 1) == 1) {
38                 y[i >> 1] = 0.0;
39             }
40         } else if (((d1 < 0.0) != (d2 < 0.0)) && ((std::abs(d1) >
41             8.9884656743115785E+307) || (std::abs(d2) >
42             8.9884656743115785E+307))) {
43             double delta1;
44             double delta2;
45             delta1 = d1 / (static_cast<double>(y.size(1)) - 1.0);
46             delta2 = d2 / (static_cast<double>(y.size(1)) - 1.0);
47             i = y.size(1);
48             for (int k = 0; k <= i - 3; k++) {
49                 y[k + 1] = (d1 + delta2 * (static_cast<double>(k) + 1.0)) - delta1 * (
50                     static_cast<double>(k) + 1.0);
51             }
52         } else {
53             double delta1;
54             delta1 = (d2 - d1) / (static_cast<double>(y.size(1)) - 1.0);
55             i = y.size(1);
56             for (int k = 0; k <= i - 3; k++) {
57                 y[k + 1] = d1 + (static_cast<double>(k) + 1.0) * delta1;
58             }
59         }
60     }
61 }
62 }
```

Here is the call graph for this function:

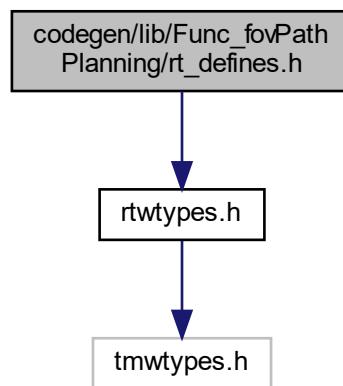


Here is the caller graph for this function:

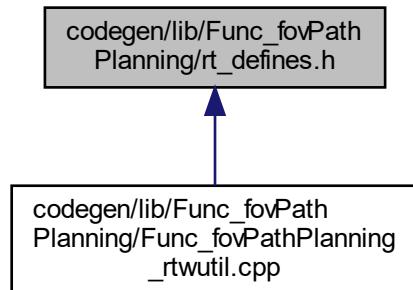


7.51 codegen/lib/Func_fovPathPlanning/rtDefines.h File Reference

```
#include "rtwtypes.h"
Include dependency graph for rtDefines.h:
```

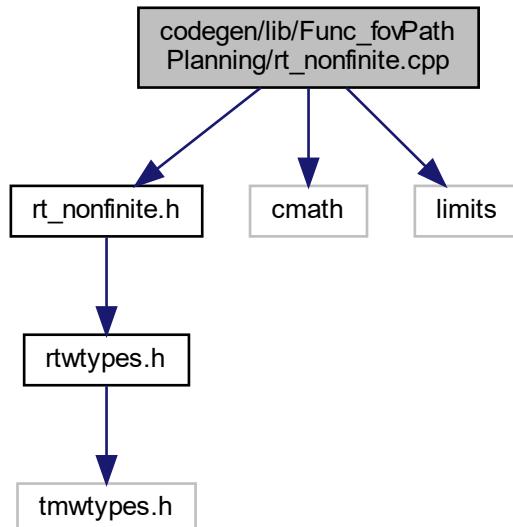


This graph shows which files directly or indirectly include this file:



7.52 codegen/lib/Func_fovPathPlanning/rt_nonfinite.cpp File Reference

```
#include "rt_nonfinite.h"
#include <cmath>
#include <limits>
Include dependency graph for rt_nonfinite.cpp:
```



Functions

- void [rt_InitInfAndNaN \(\)](#)

- boolean_T `rtIsInf` (real_T value)
- boolean_T `rtIsInfF` (real32_T value)
- boolean_T `rtIsNaN` (real_T value)
- boolean_T `rtIsNaNF` (real32_T value)

Variables

- real_T `rtInf`
- real_T `rtMinusInf`
- real_T `rtNaN`
- real32_T `rtInfF`
- real32_T `rtMinusInfF`
- real32_T `rtNaNF`

7.52.1 Function Documentation

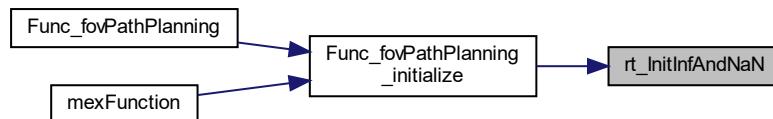
7.52.1.1 `rt_InitInfAndNaN()`

```
void rt_InitInfAndNaN ( )
```

Definition at line 33 of file `rt_nonfinite.cpp`.

```
34 {
35     rtNaN = std::numeric_limits<real_T>::quiet_NaN();
36     rtNaNF = std::numeric_limits<real32_T>::quiet_NaN();
37     rtInf = std::numeric_limits<real_T>::infinity();
38     rtInfF = std::numeric_limits<real32_T>::infinity();
39     rtMinusInf = -std::numeric_limits<real_T>::infinity();
40     rtMinusInfF = -std::numeric_limits<real32_T>::infinity();
41 }
```

Here is the caller graph for this function:



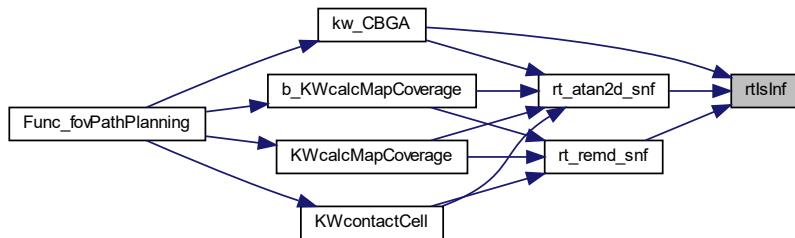
7.52.1.2 rtIsInf()

```
boolean_T rtIsInf (
    real_T value )
```

Definition at line 47 of file rt_nonfinite.cpp.

```
48 {
49     return ((value==rtInf || value==rtMinusInf) ? 1U : 0U);
50 }
```

Here is the caller graph for this function:



7.52.1.3 rtIsInfF()

```
boolean_T rtIsInfF (
    real32_T value )
```

Definition at line 56 of file rt_nonfinite.cpp.

```
57 {
58     return((value)==rtInfF || (value)==rtMinusInfF) ? 1U : 0U;
59 }
```

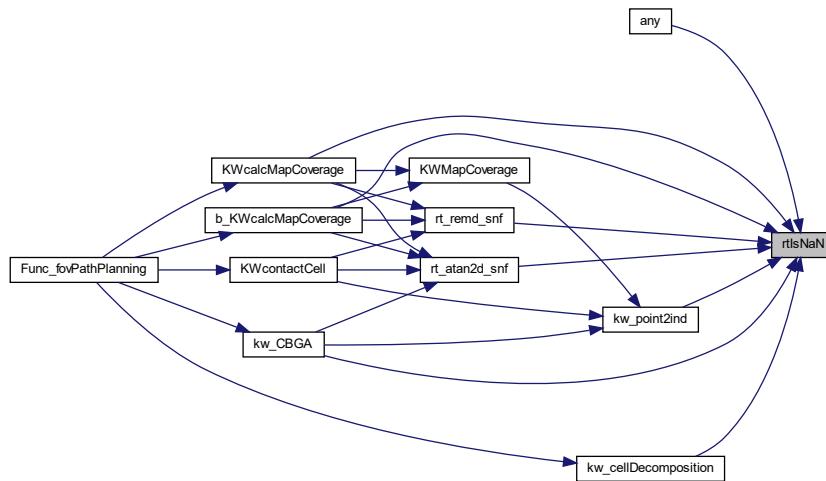
7.52.1.4 rtIsNaN()

```
boolean_T rtIsNaN (
    real_T value )
```

Definition at line 65 of file rt_nonfinite.cpp.

```
66 {
67     return ((value!=value) ? 1U : 0U);
68 }
```

Here is the caller graph for this function:



7.52.1.5 rtIsNaN()

```
boolean_T rtIsNaN (
    real32_T value )
```

Definition at line 74 of file rt_nonfinite.cpp.

```
75 {
76     return ((value!=value)? 1U : 0U);
77 }
```

7.52.2 Variable Documentation

7.52.2.1 rtInf

```
real_T rtInf
```

Definition at line 21 of file rt_nonfinite.cpp.

7.52.2.2 rtInfF

```
real32_T rtInfF
```

Definition at line 24 of file rt_nonfinite.cpp.

7.52.2.3 rtMinusInf

```
real_T rtMinusInf
```

Definition at line 22 of file rt_nonfinite.cpp.

7.52.2.4 rtMinusInff

```
real32_T rtMinusInff
```

Definition at line 25 of file rt_nonfinite.cpp.

7.52.2.5 rtNaN

```
real_T rtNaN
```

Definition at line 23 of file rt_nonfinite.cpp.

7.52.2.6 rtNaNF

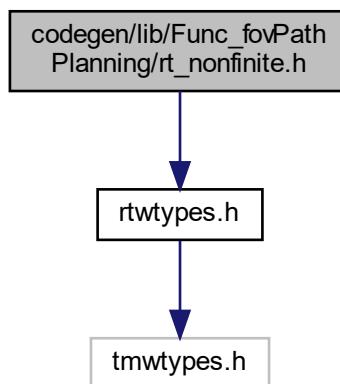
```
real32_T rtNaNF
```

Definition at line 26 of file rt_nonfinite.cpp.

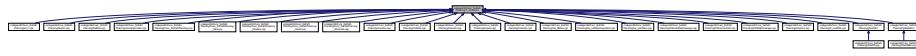
7.53 codegen/lib(Func_fovPathPlanning/rt_nonfinite.h File Reference

```
#include "rtwtypes.h"
```

Include dependency graph for rt_nonfinite.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `rt_InitInfAndNaN ()`
- boolean_T `rtIsInf (real_T value)`
- boolean_T `rtIsInfF (real32_T value)`
- boolean_T `rtIsNaN (real_T value)`
- boolean_T `rtIsNaNF (real32_T value)`

Variables

- real_T `rtInf`
- real_T `rtMinusInf`
- real_T `rtNaN`
- real32_T `rtInfF`
- real32_T `rtMinusInfF`
- real32_T `rtNaNF`

7.53.1 Function Documentation

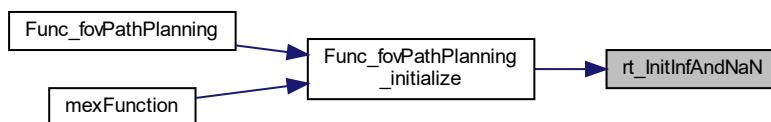
7.53.1.1 `rt_InitInfAndNaN()`

```
void rt_InitInfAndNaN ( )
```

Definition at line 33 of file `rt_nonfinite.cpp`.

```
34 {
35     rtNaN = std::numeric_limits<real_T>::quiet_NaN();
36     rtNaNF = std::numeric_limits<real32_T>::quiet_NaN();
37     rtInf = std::numeric_limits<real_T>::infinity();
38     rtInfF = std::numeric_limits<real32_T>::infinity();
39     rtMinusInf = -std::numeric_limits<real_T>::infinity();
40     rtMinusInfF = -std::numeric_limits<real32_T>::infinity();
41 }
```

Here is the caller graph for this function:



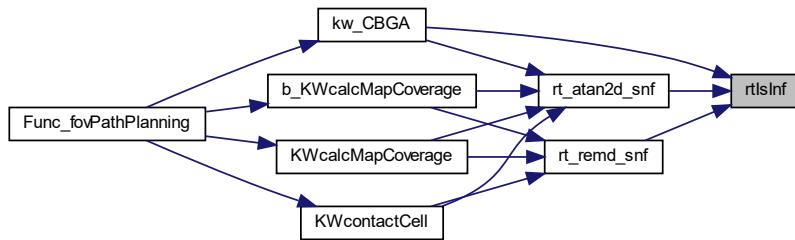
7.53.1.2 rtIsInf()

```
boolean_T rtIsInf (
    real_T value )
```

Definition at line 47 of file rt_nonfinite.cpp.

```
48 {
49     return ((value==rtInf || value==rtMinusInf) ? 1U : 0U);
50 }
```

Here is the caller graph for this function:



7.53.1.3 rtIsInfF()

```
boolean_T rtIsInfF (
    real32_T value )
```

Definition at line 56 of file rt_nonfinite.cpp.

```
57 {
58     return((value)==rtInfF || (value)==rtMinusInfF) ? 1U : 0U;
59 }
```

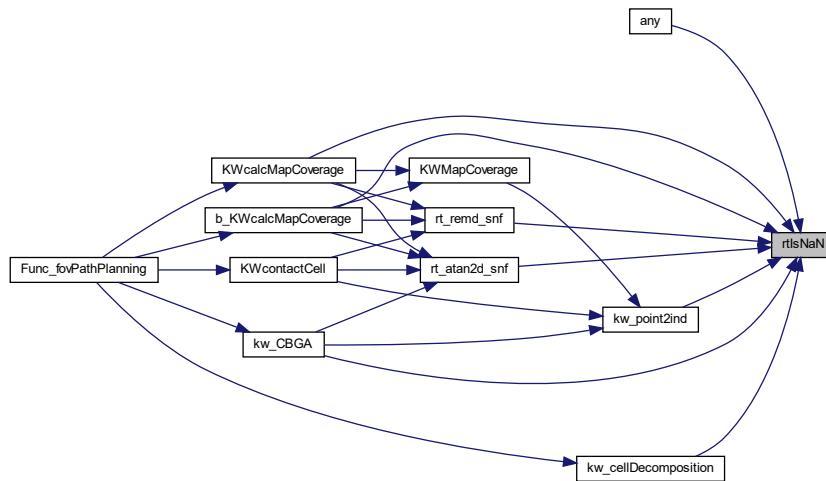
7.53.1.4 rtIsNaN()

```
boolean_T rtIsNaN (
    real_T value )
```

Definition at line 65 of file rt_nonfinite.cpp.

```
66 {
67     return ((value!=value) ? 1U : 0U);
68 }
```

Here is the caller graph for this function:



7.53.1.5 `rtIsNaN()`

```
boolean_T rtIsNaN (
    real32_T value )
```

Definition at line 74 of file `rt_nonfinite.cpp`.

```
75 {
76     return ((value!=value)? 1U : 0U);
77 }
```

7.53.2 Variable Documentation

7.53.2.1 `rtInf`

```
real_T rtInf [extern]
```

Definition at line 21 of file `rt_nonfinite.cpp`.

7.53.2.2 `rtInfF`

```
real32_T rtInfF [extern]
```

Definition at line 24 of file `rt_nonfinite.cpp`.

7.53.2.3 rtMinusInf

```
real_T rtMinusInf [extern]
```

Definition at line 22 of file rt_nonfinite.cpp.

7.53.2.4 rtMinusInfF

```
real32_T rtMinusInfF [extern]
```

Definition at line 25 of file rt_nonfinite.cpp.

7.53.2.5 rtNaN

```
real_T rtNaN [extern]
```

Definition at line 23 of file rt_nonfinite.cpp.

7.53.2.6 rtNaNF

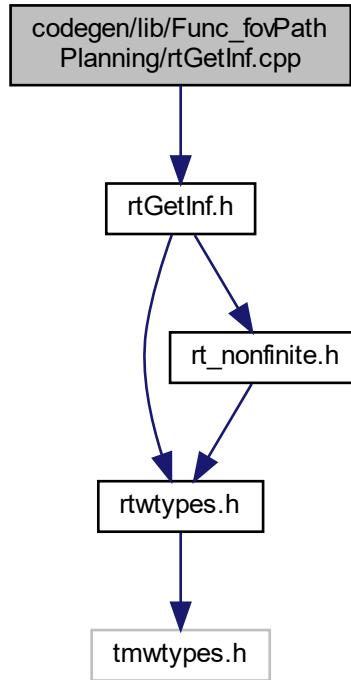
```
real32_T rtNaNF [extern]
```

Definition at line 26 of file rt_nonfinite.cpp.

7.54 codegen/lib(Func_fovPathPlanning/rtGetInf.cpp File Reference

```
#include "rtGetInf.h"
```

Include dependency graph for rtGetInf.cpp:



Functions

- real_T [rtGetInf](#) (void)
- real32_T [rtGetInfF](#) (void)
- real_T [rtGetMinusInf](#) (void)
- real32_T [rtGetMinusInfF](#) (void)

7.54.1 Function Documentation

7.54.1.1 [rtGetInf\(\)](#)

```
real_T rtGetInf (
    void )
```

Definition at line 22 of file rtGetInf.cpp.

```
23 {
24     return rtInf;
25 }
```

7.54.1.2 rtGetInfF()

```
real32_T rtGetInfF (
    void )
```

Definition at line 31 of file rtGetInf.cpp.

```
32 {
33     return rtInfF;
34 }
```

7.54.1.3 rtGetMinusInf()

```
real_T rtGetMinusInf (
    void )
```

Definition at line 40 of file rtGetInf.cpp.

```
41 {
42     return rtMinusInf;
43 }
```

7.54.1.4 rtGetMinusInfF()

```
real32_T rtGetMinusInfF (
    void )
```

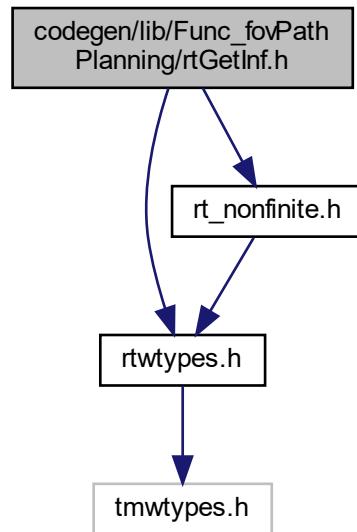
Definition at line 49 of file rtGetInf.cpp.

```
50 {
51     return rtMinusInfF;
52 }
```

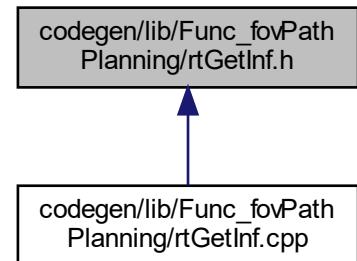
7.55 codegen/lib/Func_fovPathPlanning/rtGetInf.h File Reference

```
#include "rtwtypes.h"
#include "rt_nonfinite.h"
```

Include dependency graph for rtGetInf.h:



This graph shows which files directly or indirectly include this file:



Functions

- `real_T rtGetInf (void)`
- `real32_T rtGetInfF (void)`
- `real_T rtGetMinusInf (void)`
- `real32_T rtGetMinusInfF (void)`

7.55.1 Function Documentation

7.55.1.1 rtGetInf()

```
real_T rtGetInf (
    void )
```

Definition at line 22 of file rtGetInf.cpp.

```
23 {
24     return rtInf;
25 }
```

7.55.1.2 rtGetInfF()

```
real32_T rtGetInfF (
    void )
```

Definition at line 31 of file rtGetInf.cpp.

```
32 {
33     return rtInfF;
34 }
```

7.55.1.3 rtGetMinusInf()

```
real_T rtGetMinusInf (
    void )
```

Definition at line 40 of file rtGetInf.cpp.

```
41 {
42     return rtMinusInf;
43 }
```

7.55.1.4 rtGetMinusInfF()

```
real32_T rtGetMinusInfF (
    void )
```

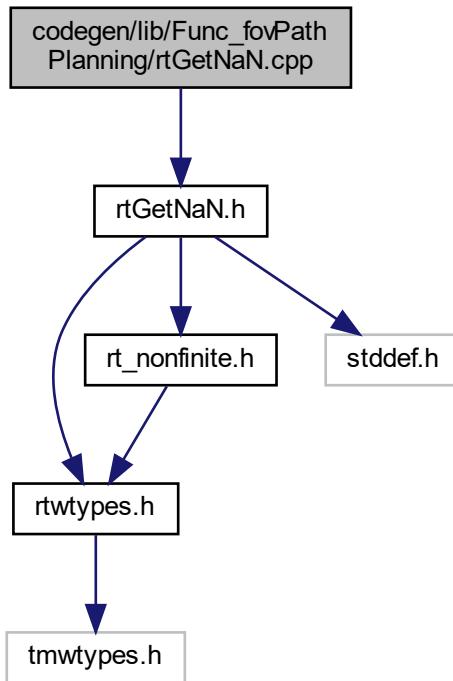
Definition at line 49 of file rtGetInf.cpp.

```
50 {
51     return rtMinusInfF;
52 }
```

7.56 codegen/lib(Func_fovPathPlanning/rtGetNaN.cpp File Reference

```
#include "rtGetNaN.h"
```

Include dependency graph for rtGetNaN.cpp:



Functions

- real_T [rtGetNaN](#) (void)
- real32_T [rtGetNaNF](#) (void)

7.56.1 Function Documentation

7.56.1.1 [rtGetNaN\(\)](#)

```
real_T rtGetNaN (
    void )
```

Definition at line 23 of file rtGetNaN.cpp.

```
24 {
25     return rtNaN;
26 }
```

7.56.1.2 rtGetNaNF()

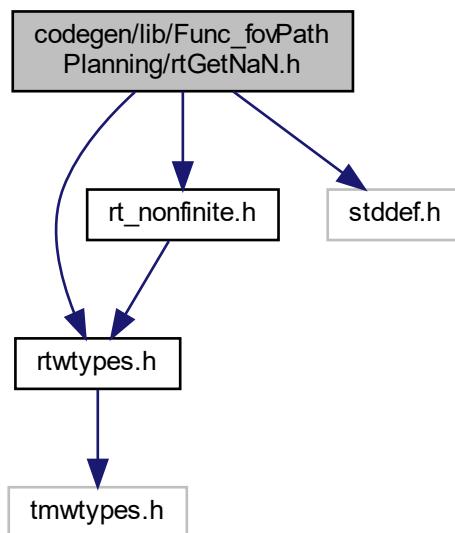
```
real32_T rtGetNaNF (
    void )
```

Definition at line 33 of file rtGetNaN.cpp.

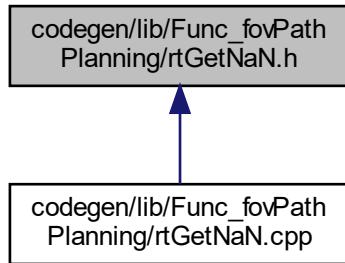
```
34 {
35     return rtNaNF;
36 }
```

7.57 codegen/lib(Func_fovPathPlanning/rtGetNaN.h File Reference

```
#include "rtwtypes.h"
#include "rt_nonfinite.h"
#include <stddef.h>
Include dependency graph for rtGetNaN.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- real_T [rtGetNaN](#) (void)
- real32_T [rtGetNaNF](#) (void)

7.57.1 Function Documentation

7.57.1.1 [rtGetNaN\(\)](#)

```
real_T rtGetNaN (
    void )
```

Definition at line 23 of file rtGetNaN.cpp.

```
24 {
25     return rtNaN;
26 }
```

7.57.1.2 [rtGetNaNF\(\)](#)

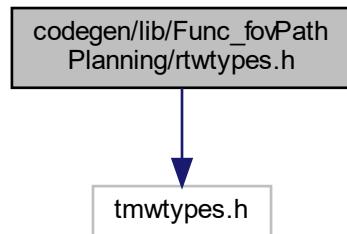
```
real32_T rtGetNaNF (
    void )
```

Definition at line 33 of file rtGetNaN.cpp.

```
34 {
35     return rtNaNF;
36 }
```

7.58 codegen/lib/Func_fovPathPlanning/rtwtypes.h File Reference

```
#include "tmwtypes.h"  
Include dependency graph for rtwtypes.h:
```



This graph shows which files directly or indirectly include this file:

