

Libembroidery

1.0.0-alpha

Generated by Doxygen 1.9.6

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 <code>_bcf_directory</code> Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 <code>dirEntries</code>	5
3.1.2.2 <code>maxNumberOfDirectoryEntries</code>	5
3.2 <code>_bcf_directory_entry</code> Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Field Documentation	6
3.2.2.1 <code>childId</code>	6
3.2.2.2 <code>CLSID</code>	6
3.2.2.3 <code>colorFlag</code>	7
3.2.2.4 <code>creationTime</code>	7
3.2.2.5 <code>directoryEntryName</code>	7
3.2.2.6 <code>directoryEntryNameLength</code>	7
3.2.2.7 <code>leftSiblingId</code>	7
3.2.2.8 <code>modifiedTime</code>	7
3.2.2.9 <code>next</code>	8
3.2.2.10 <code>objectType</code>	8
3.2.2.11 <code>rightSiblingId</code>	8
3.2.2.12 <code>startingSectorLocation</code>	8
3.2.2.13 <code>stateBits</code>	8
3.2.2.14 <code>streamSize</code>	8
3.2.2.15 <code>streamSizeHigh</code>	9
3.3 <code>_bcf_file</code> Struct Reference	9
3.3.1 Detailed Description	9
3.3.2 Field Documentation	9
3.3.2.1 <code>difat</code>	9
3.3.2.2 <code>directory</code>	9
3.3.2.3 <code>fat</code>	10
3.3.2.4 <code>header</code>	10
3.4 <code>_bcf_file_difat</code> Struct Reference	10
3.4.1 Detailed Description	10
3.4.2 Field Documentation	10
3.4.2.1 <code>fatSectorCount</code>	10
3.4.2.2 <code>fatSectorEntries</code>	11
3.4.2.3 <code>sectorSize</code>	11

3.5 _bcf_file_fat Struct Reference	11
3.5.1 Detailed Description	11
3.5.2 Field Documentation	11
3.5.2.1 fatEntries	11
3.5.2.2 fatEntryCount	12
3.5.2.3 numberOfEntriesInFatSector	12
3.6 _bcf_file_header Struct Reference	12
3.6.1 Detailed Description	12
3.6.2 Field Documentation	13
3.6.2.1 byteOrder	13
3.6.2.2 CLSID	13
3.6.2.3 firstDifatSectorLocation	13
3.6.2.4 firstDirectorySectorLocation	13
3.6.2.5 firstMiniFATSectorLocation	13
3.6.2.6 majorVersion	14
3.6.2.7 miniSectorShift	14
3.6.2.8 miniStreamCutoffSize	14
3.6.2.9 minorVersion	14
3.6.2.10 numberOfDifatSectors	14
3.6.2.11 numberOfDirectorySectors	14
3.6.2.12 numberOfFATSectors	15
3.6.2.13 numberOfMiniFatSectors	15
3.6.2.14 reserved1	15
3.6.2.15 reserved2	15
3.6.2.16 sectorShift	15
3.6.2.17 signature	15
3.6.2.18 transactionSignatureNumber	16
3.7 _vp3Hoop Struct Reference	16
3.7.1 Detailed Description	16
3.7.2 Field Documentation	16
3.7.2.1 bottom	17
3.7.2.2 bottom2	17
3.7.2.3 byte1	17
3.7.2.4 byte2	17
3.7.2.5 byte3	17
3.7.2.6 height	17
3.7.2.7 left	18
3.7.2.8 left2	18
3.7.2.9 numberOfBytesRemaining	18
3.7.2.10 numberOfColors	18
3.7.2.11 right	18
3.7.2.12 right2	18

3.7.2.13 threadLength	19
3.7.2.14 top	19
3.7.2.15 top2	19
3.7.2.16 unknown2	19
3.7.2.17 unknown3	19
3.7.2.18 unknown4	19
3.7.2.19 width	20
3.7.2.20 xOffset	20
3.7.2.21 yOffset	20
3.8 Compress Struct Reference	20
3.8.1 Detailed Description	20
3.8.2 Field Documentation	21
3.8.2.1 bit_position	21
3.8.2.2 bits_total	21
3.8.2.3 block_elements	21
3.8.2.4 character_huffman	21
3.8.2.5 character_length_huffman	21
3.8.2.6 distance_huffman	22
3.8.2.7 input_data	22
3.8.2.8 input_length	22
3.9 EmbAlignedDim_ Struct Reference	22
3.9.1 Detailed Description	22
3.9.2 Field Documentation	22
3.9.2.1 position	23
3.10 EmbAngularDim_ Struct Reference	23
3.10.1 Detailed Description	23
3.10.2 Field Documentation	23
3.10.2.1 position	23
3.11 EmbArc_ Struct Reference	23
3.11.1 Detailed Description	24
3.11.2 Field Documentation	24
3.11.2.1 end	24
3.11.2.2 mid	24
3.11.2.3 start	24
3.12 EmbArcLengthDim_ Struct Reference	24
3.12.1 Detailed Description	25
3.12.2 Field Documentation	25
3.12.2.1 position	25
3.13 EmbArray_ Struct Reference	25
3.13.1 Detailed Description	25
3.13.2 Field Documentation	25
3.13.2.1 count	26

3.13.2.2 geometry	26
3.13.2.3 length	26
3.13.2.4 stitch	26
3.13.2.5 thread	26
3.13.2.6 type	26
3.14 EmbBezier_ Struct Reference	27
3.14.1 Detailed Description	27
3.14.2 Field Documentation	27
3.14.2.1 control1	27
3.14.2.2 control2	27
3.14.2.3 end	27
3.14.2.4 start	28
3.15 EmbBlock_ Struct Reference	28
3.15.1 Detailed Description	28
3.15.2 Field Documentation	28
3.15.2.1 position	28
3.16 EmbCircle_ Struct Reference	28
3.16.1 Detailed Description	29
3.16.2 Field Documentation	29
3.16.2.1 center	29
3.16.2.2 radius	29
3.17 EmbColor_ Struct Reference	29
3.17.1 Detailed Description	29
3.17.2 Field Documentation	30
3.17.2.1 b	30
3.17.2.2 g	30
3.17.2.3 r	30
3.18 EmbDiameterDim_ Struct Reference	30
3.18.1 Detailed Description	30
3.18.2 Field Documentation	30
3.18.2.1 position	31
3.19 EmbEllipse_ Struct Reference	31
3.19.1 Detailed Description	31
3.19.2 Field Documentation	31
3.19.2.1 center	31
3.19.2.2 radius	31
3.19.2.3 rotation	32
3.20 EmbFormatList_ Struct Reference	32
3.20.1 Detailed Description	32
3.20.2 Field Documentation	32
3.20.2.1 check_for_color_file	32
3.20.2.2 color_only	32

3.20.2.3 description	33
3.20.2.4 extension	33
3.20.2.5 reader_state	33
3.20.2.6 type	33
3.20.2.7 write_external_color_file	33
3.20.2.8 writer_state	33
3.21 EmbGeometry_ Struct Reference	34
3.21.1 Detailed Description	34
3.21.2 Field Documentation	34
3.21.2.1 arc	34
3.21.2.2 circle	34
3.21.2.3 color	35
3.21.2.4 ellipse	35
3.21.2.5 flag	35
3.21.2.6 line	35
3.21.2.7 lineType	35
3.21.2.8	35
3.21.2.9 path	36
3.21.2.10 point	36
3.21.2.11 polygon	36
3.21.2.12 polyline	36
3.21.2.13 rect	36
3.21.2.14 spline	36
3.21.2.15 stitch	37
3.21.2.16 thread	37
3.21.2.17 type	37
3.21.2.18 vector	37
3.22 EmblImage_ Struct Reference	37
3.22.1 Detailed Description	38
3.22.2 Field Documentation	38
3.22.2.1 data	38
3.22.2.2 dimensions	38
3.22.2.3 height	38
3.22.2.4 name	38
3.22.2.5 path	38
3.22.2.6 position	39
3.22.2.7 width	39
3.23 EmblInfiniteLine_ Struct Reference	39
3.23.1 Detailed Description	39
3.23.2 Field Documentation	39
3.23.2.1 position	39
3.24 EmblLayer_ Struct Reference	40

3.24.1 Detailed Description	40
3.24.2 Field Documentation	40
3.24.2.1 geometry	40
3.24.2.2 name	40
3.25 EmbLeaderDim_ Struct Reference	40
3.25.1 Detailed Description	41
3.25.2 Field Documentation	41
3.25.2.1 position	41
3.26 EmbLine_ Struct Reference	41
3.26.1 Detailed Description	41
3.26.2 Field Documentation	41
3.26.2.1 color	41
3.26.2.2 end	42
3.26.2.3 lineType	42
3.26.2.4 start	42
3.27 EmbLinearDim_ Struct Reference	42
3.27.1 Detailed Description	42
3.27.2 Field Documentation	42
3.27.2.1 position	43
3.28 EmbOrdinateDim_ Struct Reference	43
3.28.1 Detailed Description	43
3.28.2 Field Documentation	43
3.28.2.1 position	43
3.29 EmbPath_ Struct Reference	43
3.29.1 Detailed Description	44
3.29.2 Field Documentation	44
3.29.2.1 color	44
3.29.2.2 flagList	44
3.29.2.3 lineType	44
3.29.2.4 pointList	44
3.30 EmbPattern_ Struct Reference	45
3.30.1 Detailed Description	45
3.30.2 Field Documentation	45
3.30.2.1 currentColorIndex	45
3.30.2.2 dstJumpsPerTrim	45
3.30.2.3 geometry	45
3.30.2.4 home	46
3.30.2.5 hoop_height	46
3.30.2.6 hoop_width	46
3.30.2.7 layer	46
3.30.2.8 stitch_list	46
3.30.2.9 thread_list	46

3.31 EmbPoint_ Struct Reference	47
3.31.1 Detailed Description	47
3.31.2 Field Documentation	47
3.31.2.1 color	47
3.31.2.2 lineType	47
3.31.2.3 position	47
3.32 EmbRadiusDim_ Struct Reference	48
3.32.1 Detailed Description	48
3.32.2 Field Documentation	48
3.32.2.1 position	48
3.33 EmbRay_ Struct Reference	48
3.33.1 Detailed Description	48
3.33.2 Field Documentation	48
3.33.2.1 position	49
3.34 EmbRect_ Struct Reference	49
3.34.1 Detailed Description	49
3.34.2 Field Documentation	49
3.34.2.1 bottom	49
3.34.2.2 left	49
3.34.2.3 radius	50
3.34.2.4 right	50
3.34.2.5 rotation	50
3.34.2.6 top	50
3.35 EmbSatinOutline_ Struct Reference	50
3.35.1 Detailed Description	50
3.35.2 Field Documentation	51
3.35.2.1 length	51
3.35.2.2 side1	51
3.35.2.3 side2	51
3.36 EmbSpline_ Struct Reference	51
3.36.1 Detailed Description	51
3.36.2 Field Documentation	51
3.36.2.1 beziers	52
3.37 EmbStitch_ Struct Reference	52
3.37.1 Detailed Description	52
3.37.2 Field Documentation	52
3.37.2.1 color	52
3.37.2.2 flags	52
3.37.2.3 x	53
3.37.2.4 y	53
3.38 EmbTextMulti_ Struct Reference	53
3.38.1 Detailed Description	53

3.38.2 Field Documentation	53
3.38.2.1 position	53
3.38.2.2 text	54
3.39 EmbTextSingle_ Struct Reference	54
3.39.1 Detailed Description	54
3.39.2 Field Documentation	54
3.39.2.1 position	54
3.39.2.2 text	54
3.40 EmbThread_ Struct Reference	55
3.40.1 Detailed Description	55
3.40.2 Field Documentation	55
3.40.2.1 catalogNumber	55
3.40.2.2 color	55
3.40.2.3 description	55
3.41 EmbTime_ Struct Reference	56
3.41.1 Detailed Description	56
3.41.2 Field Documentation	56
3.41.2.1 day	56
3.41.2.2 hour	56
3.41.2.3 minute	56
3.41.2.4 month	57
3.41.2.5 second	57
3.41.2.6 year	57
3.42 EmbVector_ Struct Reference	57
3.42.1 Detailed Description	57
3.42.2 Field Documentation	57
3.42.2.1 x	58
3.42.2.2 y	58
3.43 Huffman Struct Reference	58
3.43.1 Detailed Description	58
3.43.2 Field Documentation	58
3.43.2.1 default_value	58
3.43.2.2 lengths	59
3.43.2.3 nlengths	59
3.43.2.4 ntable	59
3.43.2.5 table	59
3.43.2.6 table_width	59
3.44 LSYSTEM Struct Reference	59
3.44.1 Detailed Description	60
3.44.2 Field Documentation	60
3.44.2.1 alphabet	60
3.44.2.2 axiom	60

3.44.2.3 constants	60
3.44.2.4 rules	60
3.45 StxThread_ Struct Reference	61
3.45.1 Detailed Description	61
3.45.2 Field Documentation	61
3.45.2.1 colorCode	61
3.45.2.2 colorName	61
3.45.2.3 sectionName	61
3.45.2.4 stxColor	62
3.45.2.5 subDescriptors	62
3.46 SubDescriptor_ Struct Reference	62
3.46.1 Detailed Description	62
3.46.2 Field Documentation	62
3.46.2.1 colorCode	62
3.46.2.2 colorName	63
3.46.2.3 someInt	63
3.46.2.4 someNum	63
3.46.2.5 someOtherInt	63
3.47 SvgAttribute_ Struct Reference	63
3.47.1 Detailed Description	63
3.47.2 Field Documentation	64
3.47.2.1 name	64
3.47.2.2 value	64
3.48 thread_color_ Struct Reference	64
3.48.1 Detailed Description	64
3.48.2 Field Documentation	64
3.48.2.1 hex_code	64
3.48.2.2 manufacturer_code	65
3.48.2.3 name	65
3.49 ThredExtension_ Struct Reference	65
3.49.1 Detailed Description	65
3.49.2 Field Documentation	65
3.49.2.1 auxFormat	65
3.49.2.2 creatorName	66
3.49.2.3 hoopX	66
3.49.2.4 hoopY	66
3.49.2.5 modifierName	66
3.49.2.6 reserved	66
3.49.2.7 stitchGranularity	66
3.50 ThredHeader_ Struct Reference	67
3.50.1 Detailed Description	67
3.50.2 Field Documentation	67

3.50.2.1 hoopSize	67
3.50.2.2 length	67
3.50.2.3 numStiches	67
3.50.2.4 reserved	68
3.50.2.5 sigVersion	68
3.51 VipHeader_ Struct Reference	68
3.51.1 Detailed Description	68
3.51.2 Field Documentation	68
3.51.2.1 attributeOffset	69
3.51.2.2 colorLength	69
3.51.2.3 magicCode	69
3.51.2.4 negativeXHoopSize	69
3.51.2.5 negativeYHoopSize	69
3.51.2.6 numberOfColors	69
3.51.2.7 numberOfStitches	70
3.51.2.8 postitiveXHoopSize	70
3.51.2.9 postitiveYHoopSize	70
3.51.2.10 stringVal	70
3.51.2.11 unknown	70
3.51.2.12 xOffset	70
3.51.2.13 yOffset	70
4 File Documentation	71
4.1 src/array.c File Reference	71
4.1.1 Function Documentation	71
4.1.1.1 embArray_addArc()	72
4.1.1.2 embArray_addCircle()	72
4.1.1.3 embArray_addEllipse()	72
4.1.1.4 embArray_addFlag()	72
4.1.1.5 embArray_addLine()	72
4.1.1.6 embArray_addPath()	73
4.1.1.7 embArray_addPoint()	73
4.1.1.8 embArray_addPolygon()	73
4.1.1.9 embArray_addPolyline()	73
4.1.1.10 embArray_addRect()	73
4.1.1.11 embArray_addStitch()	74
4.1.1.12 embArray_addVector()	74
4.1.1.13 embArray_copy()	74
4.1.1.14 embArray_create()	74
4.1.1.15 embArray_free()	74
4.1.1.16 embArray_resize()	75
4.2 array.c	75

4.3 src/compress.c File Reference	78
4.3.1 Function Documentation	79
4.3.1.1 compress_get_bits()	79
4.3.1.2 compress_get_position()	79
4.3.1.3 compress_get_token()	79
4.3.1.4 compress_init()	79
4.3.1.5 compress_load_block()	79
4.3.1.6 compress_load_character_huffman()	80
4.3.1.7 compress_load_character_length_huffman()	80
4.3.1.8 compress_load_distance_huffman()	80
4.3.1.9 compress_peek()	80
4.3.1.10 compress_pop()	80
4.3.1.11 compress_read_variable_length()	81
4.3.1.12 huffman_build_table()	81
4.3.1.13 huffman_lookup()	81
4.3.1.14 hus_compress()	81
4.3.1.15 hus_decompress()	81
4.3.2 Variable Documentation	82
4.3.2.1 huffman_lookup_data	82
4.4 compress.c	82
4.5 src/embroidery.h File Reference	85
4.5.1 Macro Definition Documentation	92
4.5.1.1 Arc_Polyester	92
4.5.1.2 Arc_Rayon	93
4.5.1.3 CHUNK_SIZE	93
4.5.1.4 CoatsAndClark_Rayon	93
4.5.1.5 dxf_color	93
4.5.1.6 EMB_ARC	93
4.5.1.7 EMB_ARRAY	93
4.5.1.8 EMB_CIRCLE	94
4.5.1.9 EMB_DIM_DIAMETER	94
4.5.1.10 EMB_DIM_LEADER	94
4.5.1.11 EMB_ELLIPSE	94
4.5.1.12 EMB_FLAG	94
4.5.1.13 EMB_FORMAT_100	94
4.5.1.14 EMB_FORMAT_100	95
4.5.1.15 EMB_FORMAT_ART	95
4.5.1.16 EMB_FORMAT_BMC	95
4.5.1.17 EMB_FORMAT_BRO	95
4.5.1.18 EMB_FORMAT_CND	95
4.5.1.19 EMB_FORMAT_COL	95
4.5.1.20 EMB_FORMAT_CSD	96

4.5.1.21 EMB_FORMAT_CSV	96
4.5.1.22 EMB_FORMAT_DAT	96
4.5.1.23 EMB_FORMAT_DEM	96
4.5.1.24 EMB_FORMAT_DSB	96
4.5.1.25 EMB_FORMAT_DST	96
4.5.1.26 EMB_FORMAT_DSZ	97
4.5.1.27 EMB_FORMAT_DXF	97
4.5.1.28 EMB_FORMAT_EDR	97
4.5.1.29 EMB_FORMAT_EMD	97
4.5.1.30 EMB_FORMAT_EXP	97
4.5.1.31 EMB_FORMAT_EXY	97
4.5.1.32 EMB_FORMAT_EYS	98
4.5.1.33 EMB_FORMAT_FXY	98
4.5.1.34 EMB_FORMAT_GC	98
4.5.1.35 EMB_FORMAT_GNC	98
4.5.1.36 EMB_FORMAT_GT	98
4.5.1.37 EMB_FORMAT_HUS	98
4.5.1.38 EMB_FORMAT_INB	99
4.5.1.39 EMB_FORMAT_INF	99
4.5.1.40 EMB_FORMAT_JEF	99
4.5.1.41 EMB_FORMAT_KSM	99
4.5.1.42 EMB_FORMAT_MAX	99
4.5.1.43 EMB_FORMAT_MIT	99
4.5.1.44 EMB_FORMAT_NEW	100
4.5.1.45 EMB_FORMAT_OFM	100
4.5.1.46 EMB_FORMAT_PCD	100
4.5.1.47 EMB_FORMAT_PCM	100
4.5.1.48 EMB_FORMAT_PCQ	100
4.5.1.49 EMB_FORMAT_PCS	100
4.5.1.50 EMB_FORMAT_PEC	101
4.5.1.51 EMB_FORMAT_PEL	101
4.5.1.52 EMB_FORMAT_PEM	101
4.5.1.53 EMB_FORMAT_PES	101
4.5.1.54 EMB_FORMAT_PHB	101
4.5.1.55 EMB_FORMAT_PHC	101
4.5.1.56 EMB_FORMAT_PLT	102
4.5.1.57 EMB_FORMAT_RGB	102
4.5.1.58 EMB_FORMAT_SEW	102
4.5.1.59 EMB_FORMAT_SHV	102
4.5.1.60 EMB_FORMAT_SST	102
4.5.1.61 EMB_FORMAT_STX	102
4.5.1.62 EMB_FORMAT_SVG	103

4.5.1.63 EMB_FORMAT_T01	103
4.5.1.64 EMB_FORMAT_T09	103
4.5.1.65 EMB_FORMAT_TAP	103
4.5.1.66 EMB_FORMAT_THR	103
4.5.1.67 EMB_FORMAT_TXT	103
4.5.1.68 EMB_FORMAT_U00	104
4.5.1.69 EMB_FORMAT_U01	104
4.5.1.70 EMB_FORMAT_VIP	104
4.5.1.71 EMB_FORMAT_VP3	104
4.5.1.72 EMB_FORMAT_XXX	104
4.5.1.73 EMB_FORMAT_ZSK	104
4.5.1.74 EMB_IMAGE	105
4.5.1.75 EMB_LINE	105
4.5.1.76 EMB_MAX_LAYERS	105
4.5.1.77 EMB_PATH	105
4.5.1.78 EMB_POINT	105
4.5.1.79 EMB_POLYGON	105
4.5.1.80 EMB_POLYLINE	106
4.5.1.81 EMB_PUBLIC	106
4.5.1.82 EMB_RECT	106
4.5.1.83 EMB_SPLINE	106
4.5.1.84 EMB_STITCH	106
4.5.1.85 EMB_TEXT_MULTI	106
4.5.1.86 EMB_TEXT_SINGLE	107
4.5.1.87 EMB_THREAD	107
4.5.1.88 EMB_VECTOR	107
4.5.1.89 EMBFORMAT_MAXDESC	107
4.5.1.90 EMBFORMAT_MAXEXT	107
4.5.1.91 EMBFORMAT_OBJECTONLY	107
4.5.1.92 EMBFORMAT_STCHANDOBJ	108
4.5.1.93 EMBFORMAT_STITCHONLY	108
4.5.1.94 EMBFORMAT_UNSUPPORTED	108
4.5.1.95 END	108
4.5.1.96 Exquisite_Polyester	108
4.5.1.97 Fufu_Polyester	108
4.5.1.98 Fufu_Rayon	109
4.5.1.99 Hemingworth_Polyester	109
4.5.1.100 hus_thread	109
4.5.1.101 Isacord_Polyester	109
4.5.1.102 Isafil_Rayon	109
4.5.1.103 jef_thread	109
4.5.1.104 JUMP	110

4.5.1.105 LIBEMBROIDERY_EMBEDDED_VERSION	110
4.5.1.106 Madeira_Polyester	110
4.5.1.107 Madeira_Rayon	110
4.5.1.108 Marathon_Polyester	110
4.5.1.109 Marathon_Rayon	110
4.5.1.110 MAX_STITCHES	111
4.5.1.111 MAX_THREADS	111
4.5.1.112 Metro_Polyester	111
4.5.1.113 NORMAL	111
4.5.1.114 numberFormats	111
4.5.1.115 Pantone	111
4.5.1.116 pcm_thread	112
4.5.1.117 pec_thread	112
4.5.1.118 RobisonAnton_Polyester	112
4.5.1.119 RobisonAnton_Rayon	112
4.5.1.120 SEQUIN	112
4.5.1.121 shv_thread	112
4.5.1.122 Sigma_Polyester	113
4.5.1.123 STOP	113
4.5.1.124 Sulky_Rayon	113
4.5.1.125 SVG_Colors	113
4.5.1.126 ThreadArt_Polyester	113
4.5.1.127 ThreadArt_Rayon	113
4.5.1.128 ThreaDelight_Polyester	114
4.5.1.129 TRIM	114
4.5.1.130 Z102_Isacord_Polyester	114
4.5.2 Typedef Documentation	114
4.5.2.1 EmbAlignedDim	114
4.5.2.2 EmbAngularDim	114
4.5.2.3 EmbArc	114
4.5.2.4 EmbArcLengthDim	115
4.5.2.5 EmbArray	115
4.5.2.6 EmbBezier	115
4.5.2.7 EmbBlock	115
4.5.2.8 EmbCircle	115
4.5.2.9 EmbColor	115
4.5.2.10 EmbDiameterDim	115
4.5.2.11 EmbEllipse	116
4.5.2.12 EmbFlag	116
4.5.2.13 EmbFormatList	116
4.5.2.14 EmbGeometry	116
4.5.2.15 EmblImage	116

4.5.2.16 EmbInfiniteLine	116
4.5.2.17 EmbLayer	116
4.5.2.18 EmbLeaderDim	117
4.5.2.19 EmbLine	117
4.5.2.20 EmbLinearDim	117
4.5.2.21 EmbOrdinateDim	117
4.5.2.22 EmbPath	117
4.5.2.23 EmbPattern	117
4.5.2.24 EmbPoint	117
4.5.2.25 EmbPolygon	118
4.5.2.26 EmbPolyline	118
4.5.2.27 EmbRadiusDim	118
4.5.2.28 EmbRay	118
4.5.2.29 EmbReal	118
4.5.2.30 EmbRect	118
4.5.2.31 EmbSatinOutline	118
4.5.2.32 EmbSpline	119
4.5.2.33 EmbStitch	119
4.5.2.34 EmbTextMulti	119
4.5.2.35 EmbTextSingle	119
4.5.2.36 EmbThread	119
4.5.2.37 EmbTime	119
4.5.2.38 EmbVector	119
4.5.2.39 L_system	120
4.5.2.40 thread_color	120
4.5.3 Function Documentation	120
4.5.3.1 convert()	120
4.5.3.2 degrees()	120
4.5.3.3 emb_identify_format()	120
4.5.3.4 emb_round()	120
4.5.3.5 embArc_clockwise()	121
4.5.3.6 embArc_init()	121
4.5.3.7 embArray_addArc()	121
4.5.3.8 embArray_addCircle()	121
4.5.3.9 embArray_addEllipse()	121
4.5.3.10 embArray_addFlag()	121
4.5.3.11 embArray_addLine()	122
4.5.3.12 embArray_addPath()	122
4.5.3.13 embArray_addPoint()	122
4.5.3.14 embArray_addPolygon()	122
4.5.3.15 embArray_addPolyline()	122
4.5.3.16 embArray_addRect()	123

4.5.3.17 embArray_addStitch()	123
4.5.3.18 embArray_addThread()	123
4.5.3.19 embArray_addVector()	123
4.5.3.20 embArray_copy()	123
4.5.3.21 embArray_create()	124
4.5.3.22 embArray_free()	124
4.5.3.23 embArray_resize()	124
4.5.3.24 embCircle_init()	124
4.5.3.25 embColor_create()	124
4.5.3.26 embColor_distance()	124
4.5.3.27 embColor_fromHexStr()	125
4.5.3.28 embColor_make()	125
4.5.3.29 embEllipse_area()	125
4.5.3.30 embEllipse_diameterX()	125
4.5.3.31 embEllipse_diameterY()	125
4.5.3.32 embEllipse_height()	125
4.5.3.33 embEllipse_init()	126
4.5.3.34 embEllipse_make()	126
4.5.3.35 embEllipse_perimeter()	126
4.5.3.36 embEllipse_width()	126
4.5.3.37 embGeometry_boundingRect()	126
4.5.3.38 embGeometry_free()	126
4.5.3.39 embGeometry_init()	127
4.5.3.40 embGeometry_move()	127
4.5.3.41 embGeometry_vulcanize()	127
4.5.3.42 emblImage_create()	127
4.5.3.43 emblImage_free()	127
4.5.3.44 emblImage_read()	127
4.5.3.45 emblImage_write()	128
4.5.3.46 embLine_intersectionPoint()	128
4.5.3.47 embLine_make()	128
4.5.3.48 embLine_normalVector()	128
4.5.3.49 embPattern_addCircleAbs()	128
4.5.3.50 embPattern_addEllipseAbs()	129
4.5.3.51 embPattern_addLineAbs()	129
4.5.3.52 embPattern_addPathAbs()	129
4.5.3.53 embPattern_addPointAbs()	129
4.5.3.54 embPattern_addPolygonAbs()	130
4.5.3.55 embPattern_addPolylineAbs()	130
4.5.3.56 embPattern_addRectAbs()	130
4.5.3.57 embPattern_addStitchAbs()	130
4.5.3.58 embPattern_addStitchRel()	131

4.5.3.59 embPattern_addThread()	131
4.5.3.60 embPattern_calcBoundingBox()	131
4.5.3.61 embPattern_center()	131
4.5.3.62 embPattern_changeColor()	131
4.5.3.63 embPattern_color_count()	132
4.5.3.64 embPattern_combine()	132
4.5.3.65 embPattern_combineJumpStitches()	132
4.5.3.66 embPattern_convertGeometry()	132
4.5.3.67 embPattern_copyPolylinesToStitchList()	132
4.5.3.68 embPattern_copyStitchListToPolylines()	132
4.5.3.69 embPattern_correctForMaxStitchLength()	133
4.5.3.70 embPattern_create()	133
4.5.3.71 embPattern_crossstitch()	133
4.5.3.72 embPattern_designDetails()	133
4.5.3.73 embPattern_end()	133
4.5.3.74 embPattern_fixColorCount()	134
4.5.3.75 embPattern_flip()	134
4.5.3.76 embPattern_flipHorizontal()	134
4.5.3.77 embPattern_flipVertical()	134
4.5.3.78 embPattern_free()	134
4.5.3.79 embPattern_hideStitchesOverLength()	135
4.5.3.80 embPattern_horizontal_fill()	135
4.5.3.81 embPattern_jumpStitches()	135
4.5.3.82 embPattern_lengthHistogram()	135
4.5.3.83 embPattern_loadExternalColorFile()	135
4.5.3.84 embPattern_maximumStitchLength()	136
4.5.3.85 embPattern_minimumStitchLength()	136
4.5.3.86 embPattern_movePolylinesToStitchList()	136
4.5.3.87 embPattern_moveStitchListToPolylines()	136
4.5.3.88 embPattern_read()	136
4.5.3.89 embPattern_readAuto()	136
4.5.3.90 embPattern_realStitches()	137
4.5.3.91 embPattern_render()	137
4.5.3.92 embPattern_scale()	137
4.5.3.93 embPattern_simulate()	137
4.5.3.94 embPattern_totalStitchLength()	137
4.5.3.95 embPattern_trimStitches()	137
4.5.3.96 embPattern_write()	138
4.5.3.97 embPattern_writeAuto()	138
4.5.3.98 embRect_area()	138
4.5.3.99 embRect_init()	138
4.5.3.100 embSatinOutline_generateSatinOutline()	138

4.5.3.101 embSatinOutline_renderStitches()	139
4.5.3.102 embThread_findNearestColor()	139
4.5.3.103 embThread_findNearestThread()	139
4.5.3.104 embThread_getRandom()	140
4.5.3.105 embTime_initNow()	140
4.5.3.106 embTime_time()	140
4.5.3.107 embVector_add()	140
4.5.3.108 embVector_angle()	140
4.5.3.109 embVector_average()	141
4.5.3.110 embVector_cross()	141
4.5.3.111 embVector_distance()	141
4.5.3.112 embVector_dot()	141
4.5.3.113 embVector_length()	141
4.5.3.114 embVector_multiply()	141
4.5.3.115 embVector_normalize()	142
4.5.3.116 embVector_relativeX()	142
4.5.3.117 embVector_relativeY()	142
4.5.3.118 embVector_subtract()	142
4.5.3.119 embVector_transpose_product()	142
4.5.3.120 embVector_unit()	142
4.5.3.121 full_test_matrix()	143
4.5.3.122 getArcCenter()	143
4.5.3.123 getArcDataFromBulge()	143
4.5.3.124 getCircleCircleIntersections()	143
4.5.3.125 getCircleTangentPoints()	143
4.5.3.126 hilbert_curve()	144
4.5.3.127 lindenmayer_system()	144
4.5.3.128 radians()	144
4.5.3.129 report()	144
4.5.3.130 testMain()	144
4.5.3.131 threadColor()	144
4.5.3.132 threadColorName()	145
4.5.3.133 threadColorNum()	145
4.5.4 Variable Documentation	145
4.5.4.1 _dxfColorTable	145
4.5.4.2 black_thread	145
4.5.4.3 emb_error	145
4.5.4.4 emb_verbose	146
4.5.4.5 embConstantPi	146
4.5.4.6 formatTable	146
4.5.4.7 husThreads	146
4.5.4.8 jefThreads	146

4.5.4.9 pcmThreads	146
4.5.4.10 pecThreadCount	147
4.5.4.11 pecThreads	147
4.5.4.12 shvThreadCount	147
4.5.4.13 shvThreads	147
4.5.4.14 vipDecodingTable	147
4.6 embroidery.h	148
4.7 src/embroidery_internal.h File Reference	155
4.7.1 Macro Definition Documentation	162
4.7.1.1 BULGETOCONTROL	163
4.7.1.2 BULGETOEND	163
4.7.1.3 CompoundFileSector_DIFAT_Sector	163
4.7.1.4 CompoundFileSector_EndOfChain	163
4.7.1.5 CompoundFileSector_FAT_Sector	163
4.7.1.6 CompoundFileSector_FreeSector	163
4.7.1.7 CompoundFileSector_MaxRegSector	164
4.7.1.8 CompoundFileStreamId_MaxRegularStreamId	164
4.7.1.9 CompoundFileStreamId_NoStream	164
4.7.1.10 CUBICTOCONTROL1	164
4.7.1.11 CUBICTOCONTROL2	164
4.7.1.12 CUBICTOEND	165
4.7.1.13 DXF_VERSION_2000	165
4.7.1.14 DXF_VERSION_2002	165
4.7.1.15 DXF_VERSION_2004	165
4.7.1.16 DXF_VERSION_2006	165
4.7.1.17 DXF_VERSION_2007	165
4.7.1.18 DXF_VERSION_2009	166
4.7.1.19 DXF_VERSION_2010	166
4.7.1.20 DXF_VERSION_2013	166
4.7.1.21 DXF_VERSION_R10	166
4.7.1.22 DXF_VERSION_R11	166
4.7.1.23 DXF_VERSION_R12	166
4.7.1.24 DXF_VERSION_R13	167
4.7.1.25 DXF_VERSION_R14	167
4.7.1.26 DXF_VERSION_R15	167
4.7.1.27 DXF_VERSION_R18	167
4.7.1.28 DXF_VERSION_R21	167
4.7.1.29 DXF_VERSION_R24	167
4.7.1.30 DXF_VERSION_R27	168
4.7.1.31 ELEMENT_A	168
4.7.1.32 ELEMENT_ANIMATE	168
4.7.1.33 ELEMENT_ANIMATECOLOR	168

4.7.1.34 ELEMENT_ANIMATEMOTION	168
4.7.1.35 ELEMENT_ANIMATETRANSFORM	168
4.7.1.36 ELEMENT_ANIMATION	169
4.7.1.37 ELEMENT_AUDIO	169
4.7.1.38 ELEMENT_CIRCLE	169
4.7.1.39 ELEMENT_DEFS	169
4.7.1.40 ELEMENT_DESC	169
4.7.1.41 ELEMENT_DISCARD	169
4.7.1.42 ELEMENT_ELLIPSE	170
4.7.1.43 ELEMENT_FONT	170
4.7.1.44 ELEMENT_FONT_FACE	170
4.7.1.45 ELEMENT_FONT_FACE_SRC	170
4.7.1.46 ELEMENT_FONT_FACE_URI	170
4.7.1.47 ELEMENT_FOREIGN_OBJECT	170
4.7.1.48 ELEMENT_G	171
4.7.1.49 ELEMENT_GLYPH	171
4.7.1.50 ELEMENT_HANDLER	171
4.7.1.51 ELEMENT_HKERN	171
4.7.1.52 ELEMENT_IMAGE	171
4.7.1.53 ELEMENT_LINE	171
4.7.1.54 ELEMENT_LINEAR_GRADIENT	172
4.7.1.55 ELEMENT_LISTENER	172
4.7.1.56 ELEMENT_METADATA	172
4.7.1.57 ELEMENT_MISSING_GLYPH	172
4.7.1.58 ELEMENT_MPATH	172
4.7.1.59 ELEMENT_PATH	172
4.7.1.60 ELEMENT_POLYGON	173
4.7.1.61 ELEMENT_POLYLINE	173
4.7.1.62 ELEMENT_PREFETCH	173
4.7.1.63 ELEMENT_RADIAL_GRADIENT	173
4.7.1.64 ELEMENT_RECT	173
4.7.1.65 ELEMENT_SCRIPT	173
4.7.1.66 ELEMENT_SET	174
4.7.1.67 ELEMENT_SOLID_COLOR	174
4.7.1.68 ELEMENT_STOP	174
4.7.1.69 ELEMENT_SVG	174
4.7.1.70 ELEMENT_SWITCH	174
4.7.1.71 ELEMENT_TBREAK	174
4.7.1.72 ELEMENT_TEXT	175
4.7.1.73 ELEMENT_TEXT_AREA	175
4.7.1.74 ELEMENT_TITLE	175
4.7.1.75 ELEMENT_TSPAN	175

4.7.1.76 ELEMENT_USE	175
4.7.1.77 ELEMENT_VIDEO	175
4.7.1.78 ELEMENT_XML	176
4.7.1.79 ELLIPSETOEND	176
4.7.1.80 ELLIPSETORAD	176
4.7.1.81 EMB_BIG_ENDIAN	176
4.7.1.82 EMB_INT16_BIG	176
4.7.1.83 EMB_INT16_LITTLE	176
4.7.1.84 EMB_INT32_BIG	177
4.7.1.85 EMB_INT32_LITTLE	177
4.7.1.86 EMB_LITTLE_ENDIAN	177
4.7.1.87 EMB_MAX	177
4.7.1.88 EMB_MIN	177
4.7.1.89 ENDIAN_HOST	177
4.7.1.90 GREEN_TERM_COLOR	178
4.7.1.91 HOOP_110X110	178
4.7.1.92 HOOP_126X110	178
4.7.1.93 HOOP_140X200	178
4.7.1.94 HOOP_230X200	178
4.7.1.95 HOOP_50X50	178
4.7.1.96 LINETO	179
4.7.1.97 MOVETO	179
4.7.1.98 N_PES_VERSIONS	179
4.7.1.99 ObjectTypeRootEntry	179
4.7.1.100 ObjectTypeStorage	179
4.7.1.101 ObjectTypeStream	180
4.7.1.102 ObjectTypeUnknown	180
4.7.1.103 PES0001	180
4.7.1.104 PES0020	180
4.7.1.105 PES0022	180
4.7.1.106 PES0030	181
4.7.1.107 PES0040	181
4.7.1.108 PES0050	181
4.7.1.109 PES0055	181
4.7.1.110 PES0056	181
4.7.1.111 PES0060	181
4.7.1.112 PES0070	182
4.7.1.113 PES0080	182
4.7.1.114 PES0090	182
4.7.1.115 PES0100	182
4.7.1.116 QUADTOCONTROL	182
4.7.1.117 QUADTOEND	182

4.7.1.118 RED_TERM_COLOR	183
4.7.1.119 RESET_TERM_COLOR	183
4.7.1.120 SVG_ATTRIBUTE	183
4.7.1.121 SVG_CATCH_ALL	183
4.7.1.122 SVG_CREATOR_EMBROIDERMODDER	183
4.7.1.123 SVG_CREATOR_ILLUSTRATOR	183
4.7.1.124 SVG_CREATOR_INKSCAPE	184
4.7.1.125 SVG_CREATOR_NULL	184
4.7.1.126 SVG_ELEMENT	184
4.7.1.127 SVG_EXPECT_ATTRIBUTE	184
4.7.1.128 SVG_EXPECT_ELEMENT	184
4.7.1.129 SVG_EXPECT_NULL	184
4.7.1.130 SVG_EXPECT_VALUE	185
4.7.1.131 SVG_MEDIA_PROPERTY	185
4.7.1.132 SVG_NULL	185
4.7.1.133 SVG_PROPERTY	185
4.7.1.134 YELLOW_TERM_COLOR	185
4.7.2 Typedef Documentation	185
4.7.2.1 bcf_directory	185
4.7.2.2 bcf_directory_entry	186
4.7.2.3 bcf_file	186
4.7.2.4 bcf_file_difat	186
4.7.2.5 bcf_file_fat	186
4.7.2.6 bcf_file_header	186
4.7.2.7 compress	186
4.7.2.8 huffman	186
4.7.2.9 StxThread	186
4.7.2.10 SubDescriptor	187
4.7.2.11 SvgAttribute	187
4.7.2.12 ThredExtension	187
4.7.2.13 ThredHeader	187
4.7.2.14 VipHeader	187
4.7.2.15 vp3Hoop	187
4.7.3 Enumeration Type Documentation	187
4.7.3.1 CSV_EXPECT	187
4.7.3.2 CSV_MODE	188
4.7.4 Function Documentation	188
4.7.4.1 bcf_difat_create()	188
4.7.4.2 bcf_directory_free()	188
4.7.4.3 bcf_file_difat_free()	189
4.7.4.4 bcf_file_fat_free()	189
4.7.4.5 bcf_file_free()	189

4.7.4.6 bcfFile_read()	189
4.7.4.7 bcfFileFat_create()	189
4.7.4.8 bcfFileHeader_isValid()	189
4.7.4.9 bcfFileHeader_read()	190
4.7.4.10 binaryReadString()	190
4.7.4.11 binaryReadUnicodeString()	190
4.7.4.12 binaryWriteUInt()	190
4.7.4.13 check_header_present()	190
4.7.4.14 CompoundFileDirectory()	191
4.7.4.15 CompoundFileDirectoryEntry()	191
4.7.4.16 compress_get_bits()	191
4.7.4.17 compress_get_position()	191
4.7.4.18 compress_get_token()	191
4.7.4.19 compress_load_block()	192
4.7.4.20 compress_load_character_huffman()	192
4.7.4.21 compress_load_character_length_huffman()	192
4.7.4.22 compress_load_distance_huffman()	192
4.7.4.23 compress_pop()	192
4.7.4.24 compress_read_variable_length()	193
4.7.4.25 copy_trim()	193
4.7.4.26 create_test_file_1()	193
4.7.4.27 create_test_file_2()	193
4.7.4.28 create_test_file_3()	193
4.7.4.29 decode_t01_record()	193
4.7.4.30 decode_tajima_ternary()	194
4.7.4.31 decodeNewStitch()	194
4.7.4.32 emb_optOut()	194
4.7.4.33 emb_readline()	194
4.7.4.34 embColor_read()	194
4.7.4.35 embColor_write()	195
4.7.4.36 emblnt_read()	195
4.7.4.37 emblnt_write()	195
4.7.4.38 encode_t01_record()	195
4.7.4.39 encode_tajima_ternary()	196
4.7.4.40 entriesInDifatSector()	196
4.7.4.41 fpad()	196
4.7.4.42 fread_int16()	196
4.7.4.43 fread_uint16()	196
4.7.4.44 GetFile()	197
4.7.4.45 huffman_build_table()	197
4.7.4.46 huffman_table_lookup()	197
4.7.4.47 hus_compress()	197

4.7.4.48 <code>hus_decompress()</code>	197
4.7.4.49 <code>loadFatFromSector()</code>	198
4.7.4.50 <code>mitDecodeStitch()</code>	198
4.7.4.51 <code>mitEncodeStitch()</code>	198
4.7.4.52 <code>numberOfEntriesInDifatSector()</code>	198
4.7.4.53 <code>pfaaffDecode()</code>	198
4.7.4.54 <code>pfaaffEncode()</code>	199
4.7.4.55 <code>printArcResults()</code>	199
4.7.4.56 <code>read100()</code>	199
4.7.4.57 <code>read10o()</code>	199
4.7.4.58 <code>readArt()</code>	199
4.7.4.59 <code>readBmc()</code>	200
4.7.4.60 <code>readBro()</code>	200
4.7.4.61 <code>readCnd()</code>	200
4.7.4.62 <code>readCol()</code>	200
4.7.4.63 <code>readCsd()</code>	200
4.7.4.64 <code>readCsv()</code>	200
4.7.4.65 <code>readDat()</code>	201
4.7.4.66 <code>readDem()</code>	201
4.7.4.67 <code>readDescriptions()</code>	201
4.7.4.68 <code>readDsb()</code>	201
4.7.4.69 <code>readDst()</code>	201
4.7.4.70 <code>readDsz()</code>	201
4.7.4.71 <code>readDxf()</code>	202
4.7.4.72 <code>readEdr()</code>	202
4.7.4.73 <code>readEmd()</code>	202
4.7.4.74 <code>readExp()</code>	202
4.7.4.75 <code>readExy()</code>	202
4.7.4.76 <code>readEys()</code>	202
4.7.4.77 <code>readFeatherPatterns()</code>	203
4.7.4.78 <code>readFullSector()</code>	203
4.7.4.79 <code>readFxy()</code>	203
4.7.4.80 <code>readGc()</code>	203
4.7.4.81 <code>readGnc()</code>	203
4.7.4.82 <code>readGt()</code>	203
4.7.4.83 <code>readHoopName()</code>	204
4.7.4.84 <code>readHus()</code>	204
4.7.4.85 <code>readImageString()</code>	204
4.7.4.86 <code>readInb()</code>	204
4.7.4.87 <code>readInf()</code>	204
4.7.4.88 <code>readJef()</code>	204
4.7.4.89 <code>readKsm()</code>	205

4.7.4.90 readMax()	205
4.7.4.91 readMit()	205
4.7.4.92 readMotifPatterns()	205
4.7.4.93 readNew()	205
4.7.4.94 readNextSector()	205
4.7.4.95 readOfm()	206
4.7.4.96 readPcd()	206
4.7.4.97 readPcm()	206
4.7.4.98 readPcq()	206
4.7.4.99 readPcs()	206
4.7.4.100 readPec()	206
4.7.4.101 readPecStitches()	207
4.7.4.102 readPel()	207
4.7.4.103 readPem()	207
4.7.4.104 readPes()	207
4.7.4.105 readPESHeaderV10()	207
4.7.4.106 readPESHeaderV5()	207
4.7.4.107 readPESHeaderV6()	208
4.7.4.108 readPESHeaderV7()	208
4.7.4.109 readPESHeaderV8()	208
4.7.4.110 readPESHeaderV9()	208
4.7.4.111 readPhb()	208
4.7.4.112 readPhc()	208
4.7.4.113 readPlt()	209
4.7.4.114 readProgrammableFills()	209
4.7.4.115 readRgb()	209
4.7.4.116 readSew()	209
4.7.4.117 readShv()	209
4.7.4.118 readSst()	209
4.7.4.119 readStx()	210
4.7.4.120 readSvg()	210
4.7.4.121 readT01()	210
4.7.4.122 readT09()	210
4.7.4.123 readTap()	210
4.7.4.124 readThr()	210
4.7.4.125 readThreads()	211
4.7.4.126 readTxt()	211
4.7.4.127 readU00()	211
4.7.4.128 readU01()	211
4.7.4.129 readVip()	211
4.7.4.130 readVp3()	211
4.7.4.131 readXxx()	212

4.7.4.132 readZsk()	212
4.7.4.133 safe_free()	212
4.7.4.134 stringInArray()	212
4.7.4.135 testEmbCircle()	212
4.7.4.136 testEmbCircle_2()	213
4.7.4.137 testEmbFormat()	213
4.7.4.138 testGeomArc()	213
4.7.4.139 testTangentPoints()	213
4.7.4.140 testThreadColor()	213
4.7.4.141 write100()	213
4.7.4.142 write10o()	214
4.7.4.143 write_24bit()	214
4.7.4.144 writeArt()	214
4.7.4.145 writeBmc()	214
4.7.4.146 writeBro()	214
4.7.4.147 writeCnd()	214
4.7.4.148 writeCol()	215
4.7.4.149 writeCsd()	215
4.7.4.150 writeCsv()	215
4.7.4.151 writeDat()	215
4.7.4.152 writeDem()	215
4.7.4.153 writeDsb()	215
4.7.4.154 writeDst()	216
4.7.4.155 writeDsz()	216
4.7.4.156 writeDxf()	216
4.7.4.157 writeEdr()	216
4.7.4.158 writeEmd()	216
4.7.4.159 writeExp()	216
4.7.4.160 writeExy()	217
4.7.4.161 writeEys()	217
4.7.4.162 writeFxy()	217
4.7.4.163 writeGc()	217
4.7.4.164 writeGnc()	217
4.7.4.165 writeGt()	217
4.7.4.166 writeHus()	218
4.7.4.167 writeInb()	218
4.7.4.168 writeInf()	218
4.7.4.169 writeJef()	218
4.7.4.170 writeKsm()	218
4.7.4.171 writeMax()	218
4.7.4.172 writeMit()	219
4.7.4.173 writeNew()	219

4.7.4.174 writeOfm()	219
4.7.4.175 writePcd()	219
4.7.4.176 writePcm()	219
4.7.4.177 writePcq()	219
4.7.4.178 writePcs()	220
4.7.4.179 writePec()	220
4.7.4.180 writePecStitches()	220
4.7.4.181 writePel()	220
4.7.4.182 writePem()	220
4.7.4.183 writePes()	220
4.7.4.184 writePhb()	221
4.7.4.185 writePhc()	221
4.7.4.186 writePlt()	221
4.7.4.187 writeRgb()	221
4.7.4.188 writeSew()	221
4.7.4.189 writeShv()	221
4.7.4.190 writeSst()	222
4.7.4.191 writeStx()	222
4.7.4.192 writeSvg()	222
4.7.4.193 writeT01()	222
4.7.4.194 writeT09()	222
4.7.4.195 writeTap()	222
4.7.4.196 writeThr()	223
4.7.4.197 writeTxt()	223
4.7.4.198 writeU00()	223
4.7.4.199 writeU01()	223
4.7.4.200 writeVip()	223
4.7.4.201 writeVp3()	223
4.7.4.202 writeXxx()	224
4.7.4.203 writeZsk()	224
4.7.5 Variable Documentation	224
4.7.5.1 imageWithFrame	224
4.8 embroidery_internal.h	224
4.9 src/encoding.c File Reference	231
4.9.1 Function Documentation	232
4.9.1.1 decode_t01_record()	232
4.9.1.2 decode_tajima_ternary()	232
4.9.1.3 decodeNewStitch()	232
4.9.1.4 embColor_fromHexStr()	232
4.9.1.5 emblnt_read()	233
4.9.1.6 emblnt_write()	233
4.9.1.7 encode_t01_record()	233

4.9.1.8 encode_tajima_ternary()	233
4.9.1.9 mitDecodeStitch()	233
4.9.1.10 mitEncodeStitch()	234
4.9.1.11 pfaffDecode()	234
4.9.1.12 pfaffEncode()	234
4.9.1.13 reverse_byte_order()	234
4.9.1.14 write_24bit()	234
4.10 encoding.c	235
4.11 src/fill.c File Reference	239
4.11.1 Function Documentation	240
4.11.1.1 dragon_curve()	240
4.11.1.2 embPattern_combine()	240
4.11.1.3 embPattern_convertGeometry()	241
4.11.1.4 embPattern_crossstitch()	241
4.11.1.5 embPattern_horizontal_fill()	241
4.11.1.6 embPattern_stitchArc()	241
4.11.1.7 embPattern_stitchCircle()	241
4.11.1.8 embPattern_stitchEllipse()	242
4.11.1.9 embPattern_stitchPath()	242
4.11.1.10 embPattern_stitchPolygon()	242
4.11.1.11 embPattern_stitchPolyline()	242
4.11.1.12 embPattern_stitchRect()	243
4.11.1.13 embPattern_stitchText()	243
4.11.1.14 embPolygon_reduceByDistance()	243
4.11.1.15 embPolygon_reduceByNth()	243
4.11.1.16 generate_dragon_curve()	243
4.11.1.17 hilbert_curve()	244
4.11.1.18 lindenmayer_system()	244
4.11.2 Variable Documentation	244
4.11.2.1 hilbert_curve_l_system	244
4.11.2.2 rules	244
4.12 fill.c	245
4.13 src/formats.c File Reference	257
4.13.1 Function Documentation	257
4.13.1.1 binaryWriteInt()	257
4.13.1.2 binaryWriteIntBE()	258
4.13.1.3 binaryWriteShort()	258
4.13.1.4 binaryWriteUInt()	258
4.13.1.5 binaryWriteUIntBE()	258
4.13.1.6 binaryWriteUShort()	258
4.13.1.7 binaryWriteUShortBE()	259
4.13.1.8 emb_identify_format()	259

4.13.1.9 embFormat_getExtension()	259
4.13.1.10 embPattern_read()	259
4.13.1.11 embPattern_readAuto()	259
4.13.1.12 embPattern_write()	260
4.13.1.13 embPattern_writeAuto()	260
4.13.1.14 fpad()	260
4.13.1.15 fread_int16()	260
4.13.1.16 fread_int32_be()	260
4.13.1.17 fread_uint16()	261
4.13.1.18 safe_free()	261
4.13.2 Variable Documentation	261
4.13.2.1 formatTable	261
4.13.2.2 imageWithFrame	261
4.14 formats.c	262
4.15 src/geometry.c File Reference	270
4.15.1 Function Documentation	270
4.15.1.1 embGeometry_boundingRect()	270
4.15.1.2 embGeometry_free()	271
4.15.1.3 embGeometry_init()	271
4.15.1.4 embGeometry_move()	271
4.15.1.5 embGeometry_vulcanize()	271
4.16 geometry.c	271
4.17 src/image.c File Reference	273
4.17.1 Function Documentation	274
4.17.1.1 image_diff()	274
4.17.1.2 writelImage()	274
4.18 image.c	274
4.19 src/main.c File Reference	276
4.19.1 Macro Definition Documentation	278
4.19.1.1 FLAG_CIRCLE	278
4.19.1.2 FLAG_CIRCLE_SHORT	279
4.19.1.3 FLAG_COMBINE	279
4.19.1.4 FLAG_CROSS_STITCH	279
4.19.1.5 FLAG_ELLIPSE	279
4.19.1.6 FLAG_ELLIPSE_SHORT	279
4.19.1.7 FLAG_FILL	279
4.19.1.8 FLAG_FILL_SHORT	280
4.19.1.9 FLAG_FORMATS	280
4.19.1.10 FLAG_FORMATS_SHORT	280
4.19.1.11 FLAG_FULL_TEST_SUITE	280
4.19.1.12 FLAG_HELP	280
4.19.1.13 FLAG_HELP_SHORT	280

4.19.1.14 FLAG_HILBERT_CURVE	281
4.19.1.15 FLAG_LINE	281
4.19.1.16 FLAG_LINE_SHORT	281
4.19.1.17 FLAG_POLYGON	281
4.19.1.18 FLAG_POLYGON_SHORT	281
4.19.1.19 FLAG_POLYLINE	281
4.19.1.20 FLAG_POLYLINE_SHORT	282
4.19.1.21 FLAG QUIET	282
4.19.1.22 FLAG QUIET_SHORT	282
4.19.1.23 FLAG_RENDER	282
4.19.1.24 FLAG_RENDER_SHORT	282
4.19.1.25 FLAG_SATIN	282
4.19.1.26 FLAG_SATIN_SHORT	283
4.19.1.27 FLAG_SIERPINSKI_TRIANGLE	283
4.19.1.28 FLAG_SIMULATE	283
4.19.1.29 FLAG_STITCH	283
4.19.1.30 FLAG_STITCH_SHORT	283
4.19.1.31 FLAG_TEST	283
4.19.1.32 FLAG_TO	284
4.19.1.33 FLAG_TO_SHORT	284
4.19.1.34 FLAG_VERBOSE	284
4.19.1.35 FLAG_VERBOSE_SHORT	284
4.19.1.36 FLAG_VERSION	284
4.19.1.37 FLAG_VERSION_SHORT	284
4.19.1.38 NUM_FLAGS	285
4.19.2 Function Documentation	285
4.19.2.1 bcf_difat_create()	285
4.19.2.2 bcf_directory_free()	285
4.19.2.3 bcf_file_free()	285
4.19.2.4 bcfFile_read()	285
4.19.2.5 bcfFileFat_create()	286
4.19.2.6 bcfFileHeader_read()	286
4.19.2.7 binaryReadString()	286
4.19.2.8 binaryReadUnicodeString()	286
4.19.2.9 check_header_present()	286
4.19.2.10 CompoundFileDirectory()	287
4.19.2.11 CompoundFileDirectoryEntry()	287
4.19.2.12 copy_trim()	287
4.19.2.13 emb_optOut()	287
4.19.2.14 emb_readline()	287
4.19.2.15 embArc_print()	288
4.19.2.16 embColor_distance()	288

4.19.2.17 embColor_read()	288
4.19.2.18 embColor_write()	288
4.19.2.19 embSatinOutline_generateSatinOutline()	288
4.19.2.20 embSatinOutline_renderStitches()	289
4.19.2.21 embThread_findNearestColor()	289
4.19.2.22 embThread_findNearestThread()	289
4.19.2.23 embThread_getRandom()	290
4.19.2.24 embTime_initNow()	290
4.19.2.25 embTime_time()	290
4.19.2.26 embVector_print()	290
4.19.2.27 entriesInDifatSector()	290
4.19.2.28 get_trim_bounds()	291
4.19.2.29 GetFile()	291
4.19.2.30 haveExtraDIFATSectors()	291
4.19.2.31 loadFatFromSector()	291
4.19.2.32 parseDIFATSectors()	291
4.19.2.33 parseDirectoryEntryName()	292
4.19.2.34 parseTime()	292
4.19.2.35 readFullSector()	292
4.19.2.36 readNextSector()	292
4.19.2.37 sectorSize()	292
4.19.2.38 seekToSector()	293
4.19.2.39 stringInArray()	293
4.19.2.40 write_24bit()	293
4.19.3 Variable Documentation	293
4.19.3.1 black_thread	293
4.19.3.2 difatEntriesInHeader	294
4.19.3.3 emb_error	294
4.19.3.4 emb_verbose	294
4.19.3.5 embConstantPi	294
4.19.3.6 sizeOfChainingEntryAtEndOfDifatSector	294
4.19.3.7 sizeOfDifatEntry	294
4.19.3.8 sizeOfDirectoryEntry	295
4.19.3.9 sizeOfFatEntry	295
4.19.3.10 WHITESPACE	295
4.20 main.c	295
4.21 src/pattern.c File Reference	315
4.21.1 Function Documentation	316
4.21.1.1 convert()	316
4.21.1.2 embPattern_addCircleAbs()	316
4.21.1.3 embPattern_addEllipseAbs()	317
4.21.1.4 embPattern_addLineAbs()	317

4.21.1.5 embPattern_addPathAbs()	317
4.21.1.6 embPattern_addPointAbs()	317
4.21.1.7 embPattern_addPolygonAbs()	318
4.21.1.8 embPattern_addPolylineObjectAbs()	318
4.21.1.9 embPattern_addRectAbs()	318
4.21.1.10 embPattern_addStitchAbs()	318
4.21.1.11 embPattern_addStitchRel()	319
4.21.1.12 embPattern_addThread()	319
4.21.1.13 embPattern_calcBoundingBox()	319
4.21.1.14 embPattern_center()	319
4.21.1.15 embPattern_changeColor()	319
4.21.1.16 embPattern_color_count()	320
4.21.1.17 embPattern_combineJumpStitches()	320
4.21.1.18 embPattern_copyPolylinesTostitch_list()	320
4.21.1.19 embPattern_copystitch_listToPolylines()	320
4.21.1.20 embPattern_correctForMaxStitchLength()	320
4.21.1.21 embPattern_create()	321
4.21.1.22 embPattern_designDetails()	321
4.21.1.23 embPattern_end()	321
4.21.1.24 embPattern_fixColorCount()	321
4.21.1.25 embPattern_flip()	321
4.21.1.26 embPattern_flipHorizontal()	322
4.21.1.27 embPattern_flipVertical()	322
4.21.1.28 embPattern_free()	322
4.21.1.29 embPattern_hideStitchesOverLength()	322
4.21.1.30 embPattern_jumpStitches()	322
4.21.1.31 embPattern_lengthHistogram()	323
4.21.1.32 embPattern_loadExternalColorFile()	323
4.21.1.33 embPattern_maximumStitchLength()	323
4.21.1.34 embPattern_minimumStitchLength()	323
4.21.1.35 embPattern_movePolylinesTostitch_list()	323
4.21.1.36 embPattern_movestitch_listToPolylines()	324
4.21.1.37 embPattern_realStitches()	324
4.21.1.38 embPattern_scale()	324
4.21.1.39 embPattern_totalStitchLength()	324
4.21.1.40 embPattern_trimStitches()	324
4.22 pattern.c	325
4.23 src/thread-color.c File Reference	339
4.23.1 Function Documentation	339
4.23.1.1 threadColor()	340
4.23.1.2 threadColorName()	340
4.23.1.3 threadColorNum()	340

4.23.2 Variable Documentation	340
4.23.2.1 _dxfColorTable	340
4.23.2.2 brand_codes	340
4.23.2.3 brand_codes_files	341
4.23.2.4 husThreads	341
4.23.2.5 jefThreads	341
4.23.2.6 pcmThreads	341
4.23.2.7 pecThreadCount	342
4.23.2.8 pecThreads	342
4.23.2.9 shvThreadCount	342
4.23.2.10 shvThreads	342
4.24 thread-color.c	342
Index	391

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

_bcf_directory	5
_bcf_directory_entry	6
_bcf_file	9
_bcf_file_difat	10
_bcf_file_fat	11
_bcf_file_header	12
_vp3Hoop	16
Compress	20
EmbAlignedDim_	22
EmbAngularDim_	23
EmbArc_	23
EmbArcLengthDim_	24
EmbArray_	25
EmbBezier_	27
EmbBlock_	28
EmbCircle_	28
EmbColor_	29
EmbDiameterDim_	30
EmbEllipse_	31
EmbFormatList_	32
EmbGeometry_	34
EmblImage_	37
EmblInfiniteLine_	39
EmbLayer_	40
EmbLeaderDim_	40
EmbLine_	41
EmbLinearDim_	42
EmbOrdinateDim_	43
EmbPath_	43
EmbPattern_	45
EmbPoint_	47
EmbRadiusDim_	48
EmbRay_	48
EmbRect_	49
EmbSatinOutline_	50

EmbSpline_	51
EmbStitch_	52
EmbTextMulti_	53
EmbTextSingle_	54
EmbThread_	55
EmbTime_	56
EmbVector_	57
Huffman	58
LSYSTEM	59
StxThread_	61
SubDescriptor_	62
SvgAttribute_	63
thread_color_	64
ThredExtension_	65
ThredHeader_	67
VipHeader_	68

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/array.c	71
src/compress.c	78
src/embroidery.h	85
src/embroidery_internal.h	155
src/encoding.c	231
src/fill.c	239
src/formats.c	257
src/geometry.c	270
src/image.c	273
src/main.c	276
src/pattern.c	315
src/thread-color.c	339

Chapter 3

Data Structure Documentation

3.1 `_bcf_directory` Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- `bcf_directory_entry` * `dirEntries`
- unsigned int `maxNumberOfDirectoryEntries`

3.1.1 Detailed Description

Definition at line 214 of file [embroidery_internal.h](#).

3.1.2 Field Documentation

3.1.2.1 `dirEntries`

```
bcf_directory_entry* dirEntries
```

Definition at line 216 of file [embroidery_internal.h](#).

3.1.2.2 `maxNumberOfDirectoryEntries`

```
unsigned int maxNumberOfDirectoryEntries
```

Definition at line 217 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.2 _bcf_directory_entry Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- char `directoryEntryName` [32]
- unsigned short `directoryEntryNameLength`
- unsigned char `objectType`
- unsigned char `colorFlag`
- unsigned int `leftSiblingId`
- unsigned int `rightSiblingId`
- unsigned int `childId`
- unsigned char `CLSID` [16]
- unsigned int `stateBits`
- `EmbTime` `creationTime`
- `EmbTime` `modifiedTime`
- unsigned int `startingSectorLocation`
- unsigned long `streamSize`
- unsigned int `streamSizeHigh`
- struct `_bcf_directory_entry` * `next`

3.2.1 Detailed Description

Definition at line 195 of file [embroidery_internal.h](#).

3.2.2 Field Documentation

3.2.2.1 childId

```
unsigned int childID
```

Definition at line 203 of file [embroidery_internal.h](#).

3.2.2.2 CLSID

```
unsigned char CLSID[16]
```

Definition at line 204 of file [embroidery_internal.h](#).

3.2.2.3 colorFlag

```
unsigned char colorFlag
```

Definition at line 200 of file [embroidery_internal.h](#).

3.2.2.4 creationTime

```
EmbTime creationTime
```

Definition at line 206 of file [embroidery_internal.h](#).

3.2.2.5 directoryEntryName

```
char directoryEntryName[32]
```

Definition at line 197 of file [embroidery_internal.h](#).

3.2.2.6 directoryEntryNameLength

```
unsigned short directoryEntryNameLength
```

Definition at line 198 of file [embroidery_internal.h](#).

3.2.2.7 leftSiblingId

```
unsigned int leftSiblingId
```

Definition at line 201 of file [embroidery_internal.h](#).

3.2.2.8 modifiedTime

```
EmbTime modifiedTime
```

Definition at line 207 of file [embroidery_internal.h](#).

3.2.2.9 next

```
struct _bcf_directory_entry* next
```

Definition at line 211 of file [embroidery_internal.h](#).

3.2.2.10 objectType

```
unsigned char objectType
```

Definition at line 199 of file [embroidery_internal.h](#).

3.2.2.11 rightSiblingId

```
unsigned int rightSiblingId
```

Definition at line 202 of file [embroidery_internal.h](#).

3.2.2.12 startingSectorLocation

```
unsigned int startingSectorLocation
```

Definition at line 208 of file [embroidery_internal.h](#).

3.2.2.13 stateBits

```
unsigned int stateBits
```

Definition at line 205 of file [embroidery_internal.h](#).

3.2.2.14 streamSize

```
unsigned long streamSize
```

Definition at line 209 of file [embroidery_internal.h](#).

3.2.2.15 streamSizeHigh

```
unsigned int streamSizeHigh
```

Definition at line 210 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.3 _bcf_file Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- [bcf_file_header](#) header
- [bcf_file_difat](#) * difat
- [bcf_file_fat](#) * fat
- [bcf_directory](#) * directory

3.3.1 Detailed Description

Definition at line 244 of file [embroidery_internal.h](#).

3.3.2 Field Documentation

3.3.2.1 difat

```
bcf\_file\_difat* difat
```

The header for the CompoundFile

Definition at line 247 of file [embroidery_internal.h](#).

3.3.2.2 directory

```
bcf\_directory* directory
```

The File Allocation Table for the Compound File

Definition at line 249 of file [embroidery_internal.h](#).

3.3.2.3 fat

```
bcf_file_fat* fat
```

The "Double Indirect FAT" for the CompoundFile

Definition at line 248 of file [embroidery_internal.h](#).

3.3.2.4 header

```
bcf_file_header header
```

Definition at line 246 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.4 _bcf_file_difat Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- unsigned int **fatSectorCount**
- unsigned int **fatSectorEntries** [109]
- unsigned int **sectorSize**

3.4.1 Detailed Description

Definition at line 181 of file [embroidery_internal.h](#).

3.4.2 Field Documentation

3.4.2.1 fatSectorCount

```
unsigned int fatSectorCount
```

Definition at line 183 of file [embroidery_internal.h](#).

3.4.2.2 fatSectorEntries

```
unsigned int fatSectorEntries[109]
```

Definition at line 184 of file [embroidery_internal.h](#).

3.4.2.3 sectorSize

```
unsigned int sectorSize
```

Definition at line 185 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.5 _bcf_file_fat Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- int [fatEntryCount](#)
- unsigned int [fatEntries](#) [255]
- unsigned int [numberOfEntriesInFatSector](#)

3.5.1 Detailed Description

Definition at line 188 of file [embroidery_internal.h](#).

3.5.2 Field Documentation

3.5.2.1 fatEntries

```
unsigned int fatEntries[255]
```

Definition at line 191 of file [embroidery_internal.h](#).

3.5.2.2 fatEntryCount

```
int fatEntryCount
```

Definition at line 190 of file [embroidery_internal.h](#).

3.5.2.3 numberOfEntriesInFatSector

```
unsigned int numberOfEntriesInFatSector
```

Definition at line 192 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.6 _bcf_file_header Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- unsigned char [signature](#) [8]
- unsigned char [CLSID](#) [16]
- unsigned short [minorVersion](#)
- unsigned short [majorVersion](#)
- unsigned short [byteOrder](#)
- unsigned short [sectorShift](#)
- unsigned short [miniSectorShift](#)
- unsigned short [reserved1](#)
- unsigned int [reserved2](#)
- unsigned int [numberOfDirectorySectors](#)
- unsigned int [numberOfFATSectors](#)
- unsigned int [firstDirectorySectorLocation](#)
- unsigned int [transactionSignatureNumber](#)
- unsigned int [miniStreamCutoffSize](#)
- unsigned int [firstMiniFATSectorLocation](#)
- unsigned int [numberOfMiniFatSectors](#)
- unsigned int [firstDifatSectorLocation](#)
- unsigned int [numberOfDifatSectors](#)

3.6.1 Detailed Description

Definition at line 222 of file [embroidery_internal.h](#).

3.6.2 Field Documentation

3.6.2.1 byteOrder

```
unsigned short byteOrder
```

Definition at line 228 of file [embroidery_internal.h](#).

3.6.2.2 CLSID

```
unsigned char CLSID[16]
```

Definition at line 225 of file [embroidery_internal.h](#).

3.6.2.3 firstDifatSectorLocation

```
unsigned int firstDifatSectorLocation
```

Definition at line 240 of file [embroidery_internal.h](#).

3.6.2.4 firstDirectorySectorLocation

```
unsigned int firstDirectorySectorLocation
```

Definition at line 235 of file [embroidery_internal.h](#).

3.6.2.5 firstMiniFATSectorLocation

```
unsigned int firstMiniFATSectorLocation
```

Definition at line 238 of file [embroidery_internal.h](#).

3.6.2.6 majorVersion

```
unsigned short majorVersion
```

Definition at line [227](#) of file [embroidery_internal.h](#).

3.6.2.7 miniSectorShift

```
unsigned short miniSectorShift
```

Definition at line [230](#) of file [embroidery_internal.h](#).

3.6.2.8 miniStreamCutoffSize

```
unsigned int miniStreamCutoffSize
```

Definition at line [237](#) of file [embroidery_internal.h](#).

3.6.2.9 minorVersion

```
unsigned short minorVersion
```

Definition at line [226](#) of file [embroidery_internal.h](#).

3.6.2.10 numberOfDifatSectors

```
unsigned int numberOfDifatSectors
```

Definition at line [241](#) of file [embroidery_internal.h](#).

3.6.2.11 numberOfDirectorySectors

```
unsigned int numberOfDirectorySectors
```

Definition at line [233](#) of file [embroidery_internal.h](#).

3.6.2.12 **numberOfFATSectors**

```
unsigned int numberOfFATSectors
```

Definition at line [234](#) of file [embroidery_internal.h](#).

3.6.2.13 **numberOfMiniFatSectors**

```
unsigned int numberOfMiniFatSectors
```

Definition at line [239](#) of file [embroidery_internal.h](#).

3.6.2.14 **reserved1**

```
unsigned short reserved1
```

Definition at line [231](#) of file [embroidery_internal.h](#).

3.6.2.15 **reserved2**

```
unsigned int reserved2
```

Definition at line [232](#) of file [embroidery_internal.h](#).

3.6.2.16 **sectorShift**

```
unsigned short sectorShift
```

Definition at line [229](#) of file [embroidery_internal.h](#).

3.6.2.17 **signature**

```
unsigned char signature[8]
```

Definition at line [224](#) of file [embroidery_internal.h](#).

3.6.2.18 transactionSignatureNumber

```
unsigned int transactionSignatureNumber
```

Definition at line 236 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.7 _vp3Hoop Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- int [right](#)
- int [bottom](#)
- int [left](#)
- int [top](#)
- int [threadLength](#)
- char [unknown2](#)
- unsigned char [numberOfColors](#)
- unsigned short [unknown3](#)
- int [unknown4](#)
- int [numberOfBytesRemaining](#)
- int [xOffset](#)
- int [yOffset](#)
- unsigned char [byte1](#)
- unsigned char [byte2](#)
- unsigned char [byte3](#)
- int [right2](#)
- int [left2](#)
- int [bottom2](#)
- int [top2](#)
- int [width](#)
- int [height](#)

3.7.1 Detailed Description

Definition at line 252 of file [embroidery_internal.h](#).

3.7.2 Field Documentation

3.7.2.1 bottom

```
int bottom
```

Definition at line 255 of file [embroidery_internal.h](#).

3.7.2.2 bottom2

```
int bottom2
```

Definition at line 275 of file [embroidery_internal.h](#).

3.7.2.3 byte1

```
unsigned char byte1
```

Definition at line 268 of file [embroidery_internal.h](#).

3.7.2.4 byte2

```
unsigned char byte2
```

Definition at line 269 of file [embroidery_internal.h](#).

3.7.2.5 byte3

```
unsigned char byte3
```

Definition at line 270 of file [embroidery_internal.h](#).

3.7.2.6 height

```
int height
```

Definition at line 279 of file [embroidery_internal.h](#).

3.7.2.7 left

```
int left
```

Definition at line 256 of file [embroidery_internal.h](#).

3.7.2.8 left2

```
int left2
```

Definition at line 274 of file [embroidery_internal.h](#).

3.7.2.9 numberOfBytesRemaining

```
int numberOfBytesRemaining
```

Definition at line 263 of file [embroidery_internal.h](#).

3.7.2.10 numberOfColors

```
unsigned char numberOfColors
```

Definition at line 260 of file [embroidery_internal.h](#).

3.7.2.11 right

```
int right
```

Definition at line 254 of file [embroidery_internal.h](#).

3.7.2.12 right2

```
int right2
```

Definition at line 273 of file [embroidery_internal.h](#).

3.7.2.13 threadLength

```
int threadLength
```

Definition at line 258 of file [embroidery_internal.h](#).

3.7.2.14 top

```
int top
```

Definition at line 257 of file [embroidery_internal.h](#).

3.7.2.15 top2

```
int top2
```

Definition at line 276 of file [embroidery_internal.h](#).

3.7.2.16 unknown2

```
char unknown2
```

Definition at line 259 of file [embroidery_internal.h](#).

3.7.2.17 unknown3

```
unsigned short unknown3
```

Definition at line 261 of file [embroidery_internal.h](#).

3.7.2.18 unknown4

```
int unknown4
```

Definition at line 262 of file [embroidery_internal.h](#).

3.7.2.19 width

```
int width
```

Definition at line 278 of file [embroidery_internal.h](#).

3.7.2.20 xOffset

```
int xOffset
```

Definition at line 265 of file [embroidery_internal.h](#).

3.7.2.21 yOffset

```
int yOffset
```

Definition at line 266 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.8 Compress Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- int [bit_position](#)
- char * [input_data](#)
- int [input_length](#)
- int [bits_total](#)
- int [block_elements](#)
- [huffman character_length_huffman](#)
- [huffman character_huffman](#)
- [huffman distance_huffman](#)

3.8.1 Detailed Description

Definition at line 368 of file [embroidery_internal.h](#).

3.8.2 Field Documentation

3.8.2.1 bit_position

```
int bit_position
```

Definition at line 369 of file [embroidery_internal.h](#).

3.8.2.2 bits_total

```
int bits_total
```

Definition at line 372 of file [embroidery_internal.h](#).

3.8.2.3 block_elements

```
int block_elements
```

Definition at line 373 of file [embroidery_internal.h](#).

3.8.2.4 character_huffman

```
huffman character_huffman
```

Definition at line 375 of file [embroidery_internal.h](#).

3.8.2.5 character_length_huffman

```
huffman character_length_huffman
```

Definition at line 374 of file [embroidery_internal.h](#).

3.8.2.6 `distance_huffman`

```
huffman distance_huffman
```

Definition at line 376 of file [embroidery_internal.h](#).

3.8.2.7 `input_data`

```
char* input_data
```

Definition at line 370 of file [embroidery_internal.h](#).

3.8.2.8 `input_length`

```
int input_length
```

Definition at line 371 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.9 `EmbAlignedDim_` Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector position`

3.9.1 Detailed Description

Definition at line 219 of file [embroidery.h](#).

3.9.2 Field Documentation

3.9.2.1 position

`EmbVector` position

Definition at line 220 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.10 EmbAngularDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector` position

3.10.1 Detailed Description

Definition at line 223 of file [embroidery.h](#).

3.10.2 Field Documentation

3.10.2.1 position

`EmbVector` position

Definition at line 224 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.11 EmbArc_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- EmbVector start
- EmbVector mid
- EmbVector end

3.11.1 Detailed Description

Definition at line 326 of file [embroidery.h](#).

3.11.2 Field Documentation

3.11.2.1 end

EmbVector end

Definition at line 330 of file [embroidery.h](#).

3.11.2.2 mid

EmbVector mid

Definition at line 329 of file [embroidery.h](#).

3.11.2.3 start

EmbVector start

Definition at line 328 of file [embroidery.h](#).

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

- src/[embroidery.h](#)

3.12 EmbArcLengthDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.12.1 Detailed Description

Definition at line [227](#) of file [embroidery.h](#).

3.12.2 Field Documentation

3.12.2.1 position

[EmbVector](#) position

Definition at line [228](#) of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.13 EmbArray_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbGeometry * geometry](#)
- [EmbStitch * stitch](#)
- [EmbThread * thread](#)
- int [count](#)
- int [length](#)
- int [type](#)

3.13.1 Detailed Description

Definition at line [408](#) of file [embroidery.h](#).

3.13.2 Field Documentation

3.13.2.1 count

```
int count
```

Definition at line 412 of file [embroidery.h](#).

3.13.2.2 geometry

```
EmbGeometry* geometry
```

Definition at line 409 of file [embroidery.h](#).

3.13.2.3 length

```
int length
```

Definition at line 413 of file [embroidery.h](#).

3.13.2.4 stitch

```
EmbStitch* stitch
```

Definition at line 410 of file [embroidery.h](#).

3.13.2.5 thread

```
EmbThread* thread
```

Definition at line 411 of file [embroidery.h](#).

3.13.2.6 type

```
int type
```

Definition at line 414 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.14 EmbBezier_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector start`
- `EmbVector control1`
- `EmbVector control2`
- `EmbVector end`

3.14.1 Detailed Description

Definition at line 367 of file [embroidery.h](#).

3.14.2 Field Documentation

3.14.2.1 control1

```
EmbVector control1
```

Definition at line 369 of file [embroidery.h](#).

3.14.2.2 control2

```
EmbVector control2
```

Definition at line 370 of file [embroidery.h](#).

3.14.2.3 end

```
EmbVector end
```

Definition at line 371 of file [embroidery.h](#).

3.14.2.4 start

`EmbVector start`

Definition at line 368 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.15 EmbBlock_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.15.1 Detailed Description

Definition at line 215 of file [embroidery.h](#).

3.15.2 Field Documentation

3.15.2.1 position

`EmbVector position`

Definition at line 216 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.16 EmbCircle_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector center](#)
- [EmbReal radius](#)

3.16.1 Detailed Description

Definition at line 343 of file [embroidery.h](#).

3.16.2 Field Documentation

3.16.2.1 center

`EmbVector` center

Definition at line 345 of file [embroidery.h](#).

3.16.2.2 radius

`EmbReal` radius

Definition at line 346 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.17 EmbColor_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `unsigned char r`
- `unsigned char g`
- `unsigned char b`

3.17.1 Detailed Description

EmbColor uses the light primaries: red, green, blue in that order.

Definition at line 182 of file [embroidery.h](#).

3.17.2 Field Documentation

3.17.2.1 b

```
unsigned char b
```

Definition at line 186 of file [embroidery.h](#).

3.17.2.2 g

```
unsigned char g
```

Definition at line 185 of file [embroidery.h](#).

3.17.2.3 r

```
unsigned char r
```

Definition at line 184 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.18 EmbDiameterDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.18.1 Detailed Description

Definition at line 231 of file [embroidery.h](#).

3.18.2 Field Documentation

3.18.2.1 position

`EmbVector` position

Definition at line 232 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.19 EmbEllipse_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector center`
- `EmbVector radius`
- `EmbReal rotation`

3.19.1 Detailed Description

Definition at line 360 of file [embroidery.h](#).

3.19.2 Field Documentation

3.19.2.1 center

`EmbVector` center

Definition at line 362 of file [embroidery.h](#).

3.19.2.2 radius

`EmbVector` radius

Definition at line 363 of file [embroidery.h](#).

3.19.2.3 rotation

```
EmbReal rotation
```

Definition at line 364 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.20 EmbFormatList_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- char [extension](#) [2+EMBFORMAT_MAXEXT]
- char [description](#) [EMBFORMAT_MAXDESC]
- char [reader_state](#)
- char [writer_state](#)
- int [type](#)
- int [color_only](#)
- int [check_for_color_file](#)
- int [write_external_color_file](#)

3.20.1 Detailed Description

Definition at line 436 of file [embroidery.h](#).

3.20.2 Field Documentation

3.20.2.1 check_for_color_file

```
int check_for_color_file
```

Definition at line 444 of file [embroidery.h](#).

3.20.2.2 color_only

```
int color_only
```

Definition at line 443 of file [embroidery.h](#).

3.20.2.3 description

```
char description[EMBFORMAT_MAXDESC]
```

Definition at line 439 of file [embroidery.h](#).

3.20.2.4 extension

```
char extension[2+EMBFORMAT_MAXEXT]
```

Definition at line 438 of file [embroidery.h](#).

3.20.2.5 reader_state

```
char reader_state
```

Definition at line 440 of file [embroidery.h](#).

3.20.2.6 type

```
int type
```

Definition at line 442 of file [embroidery.h](#).

3.20.2.7 write_external_color_file

```
int write_external_color_file
```

Definition at line 445 of file [embroidery.h](#).

3.20.2.8 writer_state

```
char writer_state
```

Definition at line 441 of file [embroidery.h](#).

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

- src/[embroidery.h](#)

3.21 EmbGeometry_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- union {
 - EmbArc arc
 - EmbCircle circle
 - EmbColor color
 - EmbEllipse ellipse
 - EmbLine line
 - EmbPath path
 - EmbPoint point
 - EmbPolygon polygon
 - EmbPolyline polyline
 - EmbRect rect
 - EmbSpline spline
 - EmbVector vector}
 - object
- EmbStitch stitch
 - EmbThread thread
 - int flag
 - int type
 - int lineType

3.21.1 Detailed Description

Definition at line 385 of file [embroidery.h](#).

3.21.2 Field Documentation

3.21.2.1 arc

```
EmbArc arc
```

Definition at line 387 of file [embroidery.h](#).

3.21.2.2 circle

```
EmbCircle circle
```

Definition at line 388 of file [embroidery.h](#).

3.21.2.3 color

```
EmbColor color
```

Definition at line 389 of file [embroidery.h](#).

3.21.2.4 ellipse

```
EmbEllipse ellipse
```

Definition at line 390 of file [embroidery.h](#).

3.21.2.5 flag

```
int flag
```

Definition at line 402 of file [embroidery.h](#).

3.21.2.6 line

```
EmbLine line
```

Definition at line 391 of file [embroidery.h](#).

3.21.2.7 lineType

```
int lineType
```

Definition at line 404 of file [embroidery.h](#).

3.21.2.8

```
union { ... } object
```

3.21.2.9 path

`EmbPath` path

Definition at line 392 of file [embroidery.h](#).

3.21.2.10 point

`EmbPoint` point

Definition at line 393 of file [embroidery.h](#).

3.21.2.11 polygon

`EmbPolygon` polygon

Definition at line 394 of file [embroidery.h](#).

3.21.2.12 polyline

`EmbPolyline` polyline

Definition at line 395 of file [embroidery.h](#).

3.21.2.13 rect

`EmbRect` rect

Definition at line 396 of file [embroidery.h](#).

3.21.2.14 spline

`EmbSpline` spline

Definition at line 397 of file [embroidery.h](#).

3.21.2.15 stitch

`EmbStitch` `stitch`

Definition at line 400 of file [embroidery.h](#).

3.21.2.16 thread

`EmbThread` `thread`

Definition at line 401 of file [embroidery.h](#).

3.21.2.17 type

`int` `type`

Definition at line 403 of file [embroidery.h](#).

3.21.2.18 vector

`EmbVector` `vector`

Definition at line 398 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.22 EmblImage_ Struct Reference

`#include <embroidery.h>`

Data Fields

- [EmbVector position](#)
- [EmbVector dimensions](#)
- `unsigned char *` [data](#)
- `int` [width](#)
- `int` [height](#)
- `char` [path](#) [200]
- `char` [name](#) [200]

3.22.1 Detailed Description

Definition at line 205 of file [embroidery.h](#).

3.22.2 Field Documentation

3.22.2.1 data

```
unsigned char* data
```

Definition at line 208 of file [embroidery.h](#).

3.22.2.2 dimensions

```
EmbVector dimensions
```

Definition at line 207 of file [embroidery.h](#).

3.22.2.3 height

```
int height
```

Definition at line 210 of file [embroidery.h](#).

3.22.2.4 name

```
char name[200]
```

Definition at line 212 of file [embroidery.h](#).

3.22.2.5 path

```
char path[200]
```

Definition at line 211 of file [embroidery.h](#).

3.22.2.6 position

`EmbVector` position

Definition at line 206 of file [embroidery.h](#).

3.22.2.7 width

`int` width

Definition at line 209 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.23 EmbInfiniteLine_Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.23.1 Detailed Description

Definition at line 251 of file [embroidery.h](#).

3.23.2 Field Documentation

3.23.2.1 position

`EmbVector` position

Definition at line 252 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.24 EmbLayer_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- char `name` [100]
- `EmbArray * geometry`

3.24.1 Detailed Description

Definition at line 417 of file [embroidery.h](#).

3.24.2 Field Documentation

3.24.2.1 `geometry`

```
EmbArray* geometry
```

Definition at line 420 of file [embroidery.h](#).

3.24.2.2 `name`

```
char name[100]
```

Definition at line 419 of file [embroidery.h](#).

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

- `src/embroidery.h`

3.25 EmbLeaderDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector position`

3.25.1 Detailed Description

Definition at line 235 of file [embroidery.h](#).

3.25.2 Field Documentation

3.25.2.1 position

`EmbVector` position

Definition at line 236 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.26 EmbLine_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector start`
- `EmbVector end`
- int `lineType`
- `EmbColor color`

3.26.1 Detailed Description

Definition at line 287 of file [embroidery.h](#).

3.26.2 Field Documentation

3.26.2.1 color

`EmbColor` color

Definition at line 292 of file [embroidery.h](#).

3.26.2.2 end

```
EmbVector end
```

Definition at line 290 of file [embroidery.h](#).

3.26.2.3 lineType

```
int lineType
```

Definition at line 291 of file [embroidery.h](#).

3.26.2.4 start

```
EmbVector start
```

Definition at line 289 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.27 EmbLinearDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.27.1 Detailed Description

Definition at line 239 of file [embroidery.h](#).

3.27.2 Field Documentation

3.27.2.1 position

`EmbVector` position

Definition at line 240 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.28 EmbOrdinateDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector` position

3.28.1 Detailed Description

Definition at line 243 of file [embroidery.h](#).

3.28.2 Field Documentation

3.28.2.1 position

`EmbVector` position

Definition at line 244 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.29 EmbPath_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbArray * pointList`
- `EmbArray * flagList`
- `int lineType`
- `EmbColor color`

3.29.1 Detailed Description

Definition at line 295 of file [embroidery.h](#).

3.29.2 Field Documentation

3.29.2.1 color

`EmbColor color`

Definition at line 300 of file [embroidery.h](#).

3.29.2.2 flagList

`EmbArray* flagList`

Definition at line 298 of file [embroidery.h](#).

3.29.2.3 lineType

`int lineType`

Definition at line 299 of file [embroidery.h](#).

3.29.2.4 pointList

`EmbArray* pointList`

Definition at line 297 of file [embroidery.h](#).

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

- `src/embroidery.h`

3.30 EmbPattern_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- unsigned int `dstJumpsPerTrim`
- `EmbVector` `home`
- `EmbReal` `hoop_width`
- `EmbReal` `hoop_height`
- `EmbArray`* `thread_list`
- `EmbArray`* `stitch_list`
- `EmbArray`* `geometry`
- `EmbLayer` `layer` [EMB_MAX_LAYERS]
- int `currentColorIndex`

3.30.1 Detailed Description

Definition at line 423 of file `embroidery.h`.

3.30.2 Field Documentation

3.30.2.1 `currentColorIndex`

```
int currentColorIndex
```

Definition at line 433 of file `embroidery.h`.

3.30.2.2 `dstJumpsPerTrim`

```
unsigned int dstJumpsPerTrim
```

Definition at line 425 of file `embroidery.h`.

3.30.2.3 `geometry`

```
EmbArray* geometry
```

Definition at line 431 of file `embroidery.h`.

3.30.2.4 home

`EmbVector` `home`

Definition at line 426 of file [embroidery.h](#).

3.30.2.5 hoop_height

`EmbReal` `hoop_height`

Definition at line 428 of file [embroidery.h](#).

3.30.2.6 hoop_width

`EmbReal` `hoop_width`

Definition at line 427 of file [embroidery.h](#).

3.30.2.7 layer

`EmbLayer` `layer[EMB_MAX_LAYERS]`

Definition at line 432 of file [embroidery.h](#).

3.30.2.8 stitch_list

`EmbArray*` `stitch_list`

Definition at line 430 of file [embroidery.h](#).

3.30.2.9 thread_list

`EmbArray*` `thread_list`

Definition at line 429 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.31 EmbPoint_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbVector position`
- `int lineType`
- `EmbColor color`

3.31.1 Detailed Description

Definition at line 280 of file [embroidery.h](#).

3.31.2 Field Documentation

3.31.2.1 color

```
EmbColor color
```

Definition at line 284 of file [embroidery.h](#).

3.31.2.2 lineType

```
int lineType
```

Definition at line 283 of file [embroidery.h](#).

3.31.2.3 position

```
EmbVector position
```

Definition at line 282 of file [embroidery.h](#).

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

- `src/embroidery.h`

3.32 EmbRadiusDim_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.32.1 Detailed Description

Definition at line [247](#) of file [embroidery.h](#).

3.32.2 Field Documentation

3.32.2.1 position

[EmbVector](#) position

Definition at line [248](#) of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.33 EmbRay_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)

3.33.1 Detailed Description

Definition at line [255](#) of file [embroidery.h](#).

3.33.2 Field Documentation

3.33.2.1 position

`EmbVector` position

Definition at line 256 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.34 EmbRect_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbReal top](#)
- [EmbReal left](#)
- [EmbReal bottom](#)
- [EmbReal right](#)
- [EmbReal rotation](#)
- [EmbReal radius](#)

3.34.1 Detailed Description

Definition at line 333 of file [embroidery.h](#).

3.34.2 Field Documentation

3.34.2.1 bottom

`EmbReal` bottom

Definition at line 337 of file [embroidery.h](#).

3.34.2.2 left

`EmbReal` left

Definition at line 336 of file [embroidery.h](#).

3.34.2.3 radius

`EmbReal radius`

Definition at line 340 of file [embroidery.h](#).

3.34.2.4 right

`EmbReal right`

Definition at line 338 of file [embroidery.h](#).

3.34.2.5 rotation

`EmbReal rotation`

Definition at line 339 of file [embroidery.h](#).

3.34.2.6 top

`EmbReal top`

Definition at line 335 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.35 EmbSatinOutline_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `int length`
- `EmbArray * side1`
- `EmbArray * side2`

3.35.1 Detailed Description

Definition at line 353 of file [embroidery.h](#).

3.35.2 Field Documentation

3.35.2.1 length

```
int length
```

Definition at line 355 of file [embroidery.h](#).

3.35.2.2 side1

```
EmbArray* side1
```

Definition at line 356 of file [embroidery.h](#).

3.35.2.3 side2

```
EmbArray* side2
```

Definition at line 357 of file [embroidery.h](#).

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

- src/[embroidery.h](#)

3.36 EmbSpline_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbArray * beziers](#)

3.36.1 Detailed Description

Definition at line 374 of file [embroidery.h](#).

3.36.2 Field Documentation

3.36.2.1 beziers

```
EmbArray* beziers
```

Definition at line 375 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.37 EmbStitch_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- int [flags](#)
- [EmbReal x](#)
- [EmbReal y](#)
- int [color](#)

3.37.1 Detailed Description

Definition at line 303 of file [embroidery.h](#).

3.37.2 Field Documentation

3.37.2.1 color

```
int color
```

Definition at line 308 of file [embroidery.h](#).

3.37.2.2 flags

```
int flags
```

Definition at line 305 of file [embroidery.h](#).

3.37.2.3 x

`EmbReal x`

Definition at line 306 of file [embroidery.h](#).

3.37.2.4 y

`EmbReal y`

Definition at line 307 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.38 EmbTextMulti_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)
- char [text](#) [200]

3.38.1 Detailed Description

Definition at line 259 of file [embroidery.h](#).

3.38.2 Field Documentation

3.38.2.1 position

`EmbVector position`

Definition at line 260 of file [embroidery.h](#).

3.38.2.2 text

```
char text[200]
```

Definition at line 261 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.39 EmbTextSingle_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbVector position](#)
- char [text](#) [200]

3.39.1 Detailed Description

Definition at line 264 of file [embroidery.h](#).

3.39.2 Field Documentation

3.39.2.1 position

[EmbVector](#) position

Definition at line 265 of file [embroidery.h](#).

3.39.2.2 text

```
char text[200]
```

Definition at line 266 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.40 EmbThread_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- `EmbColor color`
- `char description [50]`
- `char catalogNumber [30]`

3.40.1 Detailed Description

Definition at line 312 of file [embroidery.h](#).

3.40.2 Field Documentation

3.40.2.1 catalogNumber

```
char catalogNumber[30]
```

Definition at line 316 of file [embroidery.h](#).

3.40.2.2 color

```
EmbColor color
```

Definition at line 314 of file [embroidery.h](#).

3.40.2.3 description

```
char description[50]
```

Definition at line 315 of file [embroidery.h](#).

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

- `src/embroidery.h`

3.41 EmbTime_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- unsigned int [year](#)
- unsigned int [month](#)
- unsigned int [day](#)
- unsigned int [hour](#)
- unsigned int [minute](#)
- unsigned int [second](#)

3.41.1 Detailed Description

Definition at line [269](#) of file [embroidery.h](#).

3.41.2 Field Documentation

3.41.2.1 day

```
unsigned int day
```

Definition at line [273](#) of file [embroidery.h](#).

3.41.2.2 hour

```
unsigned int hour
```

Definition at line [274](#) of file [embroidery.h](#).

3.41.2.3 minute

```
unsigned int minute
```

Definition at line [275](#) of file [embroidery.h](#).

3.41.2.4 month

```
unsigned int month
```

Definition at line 272 of file [embroidery.h](#).

3.41.2.5 second

```
unsigned int second
```

Definition at line 276 of file [embroidery.h](#).

3.41.2.6 year

```
unsigned int year
```

Definition at line 271 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.42 EmbVector_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- [EmbReal x](#)
- [EmbReal y](#)

3.42.1 Detailed Description

The basic type to represent points absolutely or represent directions.

Positive y is up, units are in mm.

Definition at line 194 of file [embroidery.h](#).

3.42.2 Field Documentation

3.42.2.1 x

`EmbReal x`

Definition at line 196 of file [embroidery.h](#).

3.42.2.2 y

`EmbReal y`

Definition at line 197 of file [embroidery.h](#).

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

- [src/embroidery.h](#)

3.43 Huffman Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- int `default_value`
- int `lengths` [1000]
- int `nlengths`
- int `table` [1000]
- int `table_width`
- int `ntable`

3.43.1 Detailed Description

Definition at line 359 of file [embroidery_internal.h](#).

3.43.2 Field Documentation

3.43.2.1 default_value

```
int default_value
```

Definition at line 360 of file [embroidery_internal.h](#).

3.43.2.2 lengths

```
int lengths[1000]
```

Definition at line 361 of file [embroidery_internal.h](#).

3.43.2.3 nlenghts

```
int nlenghts
```

Definition at line 362 of file [embroidery_internal.h](#).

3.43.2.4 ntable

```
int ntable
```

Definition at line 365 of file [embroidery_internal.h](#).

3.43.2.5 table

```
int table[1000]
```

Definition at line 363 of file [embroidery_internal.h](#).

3.43.2.6 table_width

```
int table_width
```

Definition at line 364 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.44 LSYSTEM Struct Reference

```
#include <embroidery.h>
```

Data Fields

- char `axiom`
- char * `alphabet`
- char * `constants`
- char ** `rules`

3.44.1 Detailed Description

Definition at line 378 of file [embroidery.h](#).

3.44.2 Field Documentation

3.44.2.1 `alphabet`

```
char* alphabet
```

Definition at line 380 of file [embroidery.h](#).

3.44.2.2 `axiom`

```
char axiom
```

Definition at line 379 of file [embroidery.h](#).

3.44.2.3 `constants`

```
char* constants
```

Definition at line 381 of file [embroidery.h](#).

3.44.2.4 `rules`

```
char** rules
```

Definition at line 382 of file [embroidery.h](#).

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

- src/[embroidery.h](#)

3.45 StxThread_ Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- `char * colorCode`
- `char * colorName`
- `char * sectionName`
- `SubDescriptor * subDescriptors`
- `EmbColor stxColor`

3.45.1 Detailed Description

Definition at line 311 of file [embroidery_internal.h](#).

3.45.2 Field Documentation

3.45.2.1 colorCode

```
char* colorCode
```

Definition at line 313 of file [embroidery_internal.h](#).

3.45.2.2 colorName

```
char* colorName
```

Definition at line 314 of file [embroidery_internal.h](#).

3.45.2.3 sectionName

```
char* sectionName
```

Definition at line 315 of file [embroidery_internal.h](#).

3.45.2.4 stxColor

```
EmbColor stxColor
```

Definition at line 317 of file [embroidery_internal.h](#).

3.45.2.5 subDescriptors

```
SubDescriptor* subDescriptors
```

Definition at line 316 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.46 SubDescriptor_ Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- int [someNum](#)
- int [someInt](#)
- int [someOtherInt](#)
- char * [colorCode](#)
- char * [colorName](#)

3.46.1 Detailed Description

Definition at line 302 of file [embroidery_internal.h](#).

3.46.2 Field Documentation

3.46.2.1 colorCode

```
char* colorCode
```

Definition at line 307 of file [embroidery_internal.h](#).

3.46.2.2 colorName

```
char* colorName
```

Definition at line 308 of file [embroidery_internal.h](#).

3.46.2.3 someInt

```
int someInt
```

Definition at line 305 of file [embroidery_internal.h](#).

3.46.2.4 someNum

```
int someNum
```

Definition at line 304 of file [embroidery_internal.h](#).

3.46.2.5 someOtherInt

```
int someOtherInt
```

Definition at line 306 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.47 SvgAttribute_ Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- `char * name`
- `char * value`

3.47.1 Detailed Description

Definition at line 353 of file [embroidery_internal.h](#).

3.47.2 Field Documentation

3.47.2.1 name

```
char* name
```

Definition at line 355 of file [embroidery_internal.h](#).

3.47.2.2 value

```
char* value
```

Definition at line 356 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.48 thread_color_ Struct Reference

```
#include <embroidery.h>
```

Data Fields

- char [name](#) [22]
- unsigned int [hex_code](#)
- int [manufacturer_code](#)

3.48.1 Detailed Description

Definition at line 319 of file [embroidery.h](#).

3.48.2 Field Documentation

3.48.2.1 hex_code

```
unsigned int hex_code
```

Definition at line 321 of file [embroidery.h](#).

3.48.2.2 manufacturer_code

```
int manufacturer_code
```

Definition at line 322 of file [embroidery.h](#).

3.48.2.3 name

```
char name[22]
```

Definition at line 320 of file [embroidery.h](#).

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

- src/[embroidery.h](#)

3.49 ThredExtension_ Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- float [hoopX](#)
- float [hoopY](#)
- float [stitchGranularity](#)
- char [creatorName](#) [50]
- char [modifierName](#) [50]
- char [auxFormat](#)
- char [reserved](#) [31]

3.49.1 Detailed Description

Definition at line 291 of file [embroidery_internal.h](#).

3.49.2 Field Documentation

3.49.2.1 auxFormat

```
char auxFormat
```

Definition at line 298 of file [embroidery_internal.h](#).

3.49.2.2 creatorName

```
char creatorName[50]
```

Definition at line 296 of file [embroidery_internal.h](#).

3.49.2.3 hoopX

```
float hoopX
```

Definition at line 293 of file [embroidery_internal.h](#).

3.49.2.4 hoopY

```
float hoopY
```

Definition at line 294 of file [embroidery_internal.h](#).

3.49.2.5 modifierName

```
char modifierName[50]
```

Definition at line 297 of file [embroidery_internal.h](#).

3.49.2.6 reserved

```
char reserved[31]
```

Definition at line 299 of file [embroidery_internal.h](#).

3.49.2.7 stitchGranularity

```
float stitchGranularity
```

Definition at line 295 of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

3.50 ThredHeader_ Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- unsigned int `sigVersion`
- unsigned int `length`
- unsigned short `numStiches`
- unsigned short `hoopSize`
- unsigned short `reserved` [7]

3.50.1 Detailed Description

Definition at line 282 of file [embroidery_internal.h](#).

3.50.2 Field Documentation

3.50.2.1 hoopSize

```
unsigned short hoopSize
```

Definition at line 287 of file [embroidery_internal.h](#).

3.50.2.2 length

```
unsigned int length
```

Definition at line 285 of file [embroidery_internal.h](#).

3.50.2.3 numStiches

```
unsigned short numStiches
```

Definition at line 286 of file [embroidery_internal.h](#).

3.50.2.4 reserved

```
unsigned short reserved[7]
```

Definition at line 288 of file [embroidery_internal.h](#).

3.50.2.5 sigVersion

```
unsigned int sigVersion
```

Definition at line 284 of file [embroidery_internal.h](#).

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

- src/[embroidery_internal.h](#)

3.51 VipHeader_ Struct Reference

```
#include <embroidery_internal.h>
```

Data Fields

- int [magicCode](#)
- int [numberOfStitches](#)
- int [numberOfColors](#)
- short [positiveXHoopSize](#)
- short [positiveYHoopSize](#)
- short [negativeXHoopSize](#)
- short [negativeYHoopSize](#)
- int [attributeOffset](#)
- int [xOffset](#)
- int [yOffset](#)
- unsigned char [stringVal](#) [8]
- short [unknown](#)
- int [colorLength](#)

3.51.1 Detailed Description

Definition at line 320 of file [embroidery_internal.h](#).

3.51.2 Field Documentation

3.51.2.1 attributeOffset

```
int attributeOffset
```

Definition at line 328 of file [embroidery_internal.h](#).

3.51.2.2 colorLength

```
int colorLength
```

Definition at line 333 of file [embroidery_internal.h](#).

3.51.2.3 magicCode

```
int magicCode
```

Definition at line 321 of file [embroidery_internal.h](#).

3.51.2.4 negativeXHoopSize

```
short negativeXHoopSize
```

Definition at line 326 of file [embroidery_internal.h](#).

3.51.2.5 negativeYHoopSize

```
short negativeYHoopSize
```

Definition at line 327 of file [embroidery_internal.h](#).

3.51.2.6 numberOfColors

```
int numberOfColors
```

Definition at line 323 of file [embroidery_internal.h](#).

3.51.2.7 **numberOfStitches**

```
int numberOfStitches
```

Definition at line [322](#) of file [embroidery_internal.h](#).

3.51.2.8 **postitiveXHoopSize**

```
short postitiveXHoopSize
```

Definition at line [324](#) of file [embroidery_internal.h](#).

3.51.2.9 **postitiveYHoopSize**

```
short postitiveYHoopSize
```

Definition at line [325](#) of file [embroidery_internal.h](#).

3.51.2.10 **stringVal**

```
unsigned char stringVal[8]
```

Definition at line [331](#) of file [embroidery_internal.h](#).

3.51.2.11 **unknown**

```
short unknown
```

Definition at line [332](#) of file [embroidery_internal.h](#).

3.51.2.12 **xOffset**

```
int xOffset
```

Definition at line [329](#) of file [embroidery_internal.h](#).

3.51.2.13 **yOffset**

```
int yOffset
```

Definition at line [330](#) of file [embroidery_internal.h](#).

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

- [src/embroidery_internal.h](#)

Chapter 4

File Documentation

4.1 src/array.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "embroidery_internal.h"
```

Functions

- `EmbArray * embArray_create (int type)`
- `int embArray_resize (EmbArray *a)`
- `void embArray_copy (EmbArray *dst, EmbArray *src)`
- `int embArray_addArc (EmbArray *a, EmbArc b)`
- `int embArray_addCircle (EmbArray *a, EmbCircle b)`
- `int embArray_addEllipse (EmbArray *a, EmbEllipse b)`
- `int embArray_addFlag (EmbArray *a, EmbFlag b)`
- `int embArray_addLine (EmbArray *a, EmbLine b)`
- `int embArray_addPath (EmbArray *a, EmbPath b)`
- `int embArray_addPoint (EmbArray *a, EmbPoint b)`
- `int embArray_addPolyline (EmbArray *a, EmbPolyline b)`
- `int embArray_addPolygon (EmbArray *a, EmbPolygon b)`
- `int embArray_addRect (EmbArray *a, EmbRect b)`
- `int embArray_addStitch (EmbArray *a, EmbStitch b)`
- `int embArray_addVector (EmbArray *a, EmbVector b)`
- `void embArray_free (EmbArray *a)`

4.1.1 Function Documentation

4.1.1.1 `embArray_addArc()`

```
int embArray_addArc (
    EmbArray * a,
    EmbArc b )
```

Definition at line 96 of file [array.c](#).

4.1.1.2 `embArray_addCircle()`

```
int embArray_addCircle (
    EmbArray * a,
    EmbCircle b )
```

Definition at line 108 of file [array.c](#).

4.1.1.3 `embArray_addEllipse()`

```
int embArray_addEllipse (
    EmbArray * a,
    EmbEllipse b )
```

Definition at line 120 of file [array.c](#).

4.1.1.4 `embArray_addFlag()`

```
int embArray_addFlag (
    EmbArray * a,
    EmbFlag b )
```

Definition at line 132 of file [array.c](#).

4.1.1.5 `embArray_addLine()`

```
int embArray_addLine (
    EmbArray * a,
    EmbLine b )
```

Definition at line 144 of file [array.c](#).

4.1.1.6 embArray_addPath()

```
int embArray_addPath (
    EmbArray * a,
    EmbPath b )
```

Definition at line 156 of file [array.c](#).

4.1.1.7 embArray_addPoint()

```
int embArray_addPoint (
    EmbArray * a,
    EmbPoint b )
```

Definition at line 168 of file [array.c](#).

4.1.1.8 embArray_addPolygon()

```
int embArray_addPolygon (
    EmbArray * a,
    EmbPolygon b )
```

Definition at line 192 of file [array.c](#).

4.1.1.9 embArray_addPolyline()

```
int embArray_addPolyline (
    EmbArray * a,
    EmbPolyline b )
```

Definition at line 180 of file [array.c](#).

4.1.1.10 embArray_addRect()

```
int embArray_addRect (
    EmbArray * a,
    EmbRect b )
```

Definition at line 204 of file [array.c](#).

4.1.1.11 embArray_addStitch()

```
int embArray_addStitch (
    EmbArray * a,
    EmbStitch b )
```

Definition at line 216 of file [array.c](#).

4.1.1.12 embArray_addVector()

```
int embArray_addVector (
    EmbArray * a,
    EmbVector b )
```

Definition at line 227 of file [array.c](#).

4.1.1.13 embArray_copy()

```
void embArray_copy (
    EmbArray * dst,
    EmbArray * src )
```

Definition at line 74 of file [array.c](#).

4.1.1.14 embArray_create()

```
EmbArray * embArray_create (
    int type )
```

Definition at line 19 of file [array.c](#).

4.1.1.15 embArray_free()

```
void embArray_free (
    EmbArray * a )
```

Definition at line 239 of file [array.c](#).

4.1.1.16 embArray_resize()

```
int embArray_resize (
    EmbArray * a )
```

Definition at line 41 of file [array.c](#).

4.2 array.c

[Go to the documentation of this file.](#)

```
00001 /*
00002  * This file is part of libembroidery.
00003  *
00004  * Copyright 2018-2022 The Embroidermodder Team
00005  * Licensed under the terms of the zlib license.
00006  *
00007  * -----
00008  *
00009  * The array management for libembroidery's arrays.
00010 */
00011
00012 #include <stdio.h>
00013 #include <stdlib.h>
00014 #include <string.h>
00015
00016 #include "embroidery_internal.h"
00017
00018 EmbArray*
00019 embArray_create(int type)
00020 {
00021     EmbArray *a;
00022     a = (EmbArray*)malloc(sizeof(EmbArray));
00023     a->type = type;
00024     a->length = CHUNK_SIZE;
00025     a->count = 0;
00026     switch (type) {
00027         case EMB_STITCH:
00028             a->stitch = (EmbStitch*)malloc(CHUNK_SIZE*sizeof(EmbStitch));
00029             break;
00030         case EMB_THREAD:
00031             a->thread = (EmbThread*)malloc(CHUNK_SIZE*sizeof(EmbThread));
00032             break;
00033         default:
00034             a->geometry = (EmbGeometry*)malloc(CHUNK_SIZE*sizeof(EmbGeometry));
00035             break;
00036     }
00037     return a;
00038 }
00039
00040 int
00041 embArray_resize(EmbArray *a)
00042 {
00043     if (a->count < a->length) {
00044         return 1;
00045     }
00046     a->length += CHUNK_SIZE;
00047     switch (a->type) {
00048         case EMB_STITCH:
00049             a->stitch = (EmbStitch*)realloc(a->stitch, a->length*sizeof(EmbStitch));
00050             if (!a->stitch) {
00051                 /* TODO: Error reporting */
00052                 return 0;
00053             }
00054             break;
00055         case EMB_THREAD:
00056             a->thread = (EmbThread*)realloc(a->thread, a->length*sizeof(EmbThread));
00057             if (!a->thread) {
00058                 /* TODO: Error reporting */
00059                 return 0;
00060             }
00061             break;
00062         default:
00063             a->geometry = (EmbGeometry *)realloc(a->geometry, a->length*sizeof(EmbGeometry));
00064             if (!a->geometry) {
00065                 /* TODO: Error reporting */
00066                 return 0;
00067             }
00068             break;
```

```
00069      }
00070
00071     return 1;
00072 }
00073
00074 void embArray_copy(EmbArray *dst, EmbArray *src)
00075 {
00076     dst = embArray_create(src->type);
00077     dst->length = src->length;
00078     dst->count = src->count;
00079     embArray_resize(dst);
00080     /* BUG: Potential failure to copy path memory, only copy pointers? */
00081
00082     switch (dst->type) {
00083     case EMB_STITCH:
00084         memcpy(dst->stitch, src->stitch, sizeof(EmbStitch)*src->count);
00085         break;
00086     case EMB_THREAD:
00087         memcpy(dst->thread, src->thread, sizeof(EmbThread)*src->count);
00088         break;
00089     default:
00090         memcpy(dst->geometry, src->geometry, sizeof(EmbGeometry)*src->count);
00091         break;
00092     }
00093 }
00094
00095 int
00096 embArray_addArc(EmbArray *a, EmbArc b)
00097 {
00098     a->count++;
00099     if (!embArray_resize(a)) {
00100         return 0;
00101     }
00102     a->geometry[a->count - 1].object.arc = b;
00103     a->geometry[a->count - 1].type = EMB_ARC;
00104     return 1;
00105 }
00106
00107 int
00108 embArray_addCircle(EmbArray *a, EmbCircle b)
00109 {
00110     a->count++;
00111     if (!embArray_resize(a)) {
00112         return 0;
00113     }
00114     a->geometry[a->count - 1].object.circle = b;
00115     a->geometry[a->count - 1].type = EMB_CIRCLE;
00116     return 1;
00117 }
00118
00119 int
00120 embArray_addEllipse(EmbArray *a, EmbEllipse b)
00121 {
00122     a->count++;
00123     if (!embArray_resize(a)) {
00124         return 0;
00125     }
00126     a->geometry[a->count - 1].object.ellipse = b;
00127     a->geometry[a->count - 1].type = EMB_ELLIPSE;
00128     return 1;
00129 }
00130
00131 int
00132 embArray_addFlag(EmbArray *a, EmbFlag b)
00133 {
00134     a->count++;
00135     if (!embArray_resize(a)) {
00136         return 0;
00137     }
00138     a->geometry[a->count - 1].flag = b;
00139     a->geometry[a->count - 1].type = EMB_FLAG;
00140     return 1;
00141 }
00142
00143 int
00144 embArray_addLine(EmbArray *a, EmbLine b)
00145 {
00146     a->count++;
00147     if (!embArray_resize(a)) {
00148         return 0;
00149     }
00150     a->geometry[a->count - 1].object.line = b;
00151     a->geometry[a->count - 1].type = EMB_LINE;
00152     return 1;
00153 }
00154
00155 int
```

```
00156 embArray_addPath(EmbArray *a, EmbPath b)
00157 {
00158     a->count++;
00159     if (!embArray_resize(a)) {
00160         return 0;
00161     }
00162     a->geometry[a->count - 1].object.path = b;
00163     a->geometry[a->count - 1].type = EMB_PATH;
00164     return 1;
00165 }
00166
00167 int
00168 embArray_addPoint(EmbArray *a, EmbPoint b)
00169 {
00170     a->count++;
00171     if (!embArray_resize(a)) {
00172         return 0;
00173     }
00174     a->geometry[a->count - 1].object.point = b;
00175     a->geometry[a->count - 1].type = EMB_POINT;
00176     return 1;
00177 }
00178
00179 int
00180 embArray_addPolyline(EmbArray *a, EmbPolyline b)
00181 {
00182     a->count++;
00183     if (!embArray_resize(a)) {
00184         return 0;
00185     }
00186     a->geometry[a->count - 1].object.polyline = b;
00187     a->geometry[a->count - 1].type = EMB_POLYLINE;
00188     return 1;
00189 }
00190
00191 int
00192 embArray_addPolygon(EmbArray *a, EmbPolygon b)
00193 {
00194     a->count++;
00195     if (!embArray_resize(a)) {
00196         return 0;
00197     }
00198     a->geometry[a->count - 1].object.polygon = b;
00199     a->geometry[a->count - 1].type = EMB_POLYGON;
00200     return 1;
00201 }
00202
00203 int
00204 embArray_addRect(EmbArray *a, EmbRect b)
00205 {
00206     a->count++;
00207     if (!embArray_resize(a)) {
00208         return 0;
00209     }
00210     a->geometry[a->count - 1].object.rect = b;
00211     a->geometry[a->count - 1].type = EMB_RECT;
00212     return 1;
00213 }
00214
00215 int
00216 embArray_addStitch(EmbArray *a, EmbStitch b)
00217 {
00218     a->count++;
00219     if (!embArray_resize(a)) {
00220         return 0;
00221     }
00222     a->stitch[a->count - 1] = b;
00223     return 1;
00224 }
00225
00226 int
00227 embArray_addVector(EmbArray *a, EmbVector b)
00228 {
00229     a->count++;
00230     if (!embArray_resize(a)) {
00231         return 0;
00232     }
00233     a->geometry[a->count - 1].object.vector = b;
00234     a->geometry[a->count - 1].type = EMB_VECTOR;
00235     return 1;
00236 }
00237
00238 void
00239 embArray_free(EmbArray* a)
00240 {
00241     int i;
00242     if (!a) {
```

```

00243     return;
00244 }
00245 switch (a->type) {
00246 case EMB_STITCH:
00247     free(a->stitch);
00248     break;
00249 case EMB_THREAD:
00250     free(a->thread);
00251     break;
00252 default:
00253     for (i = 0; i < a->count; i++) {
00254         EmbGeometry g = a->geometry[i];
00255         switch (a->geometry[i].type) {
00256             case EMB_PATH: {
00257                 embArray_free(g.object.path.pointList);
00258                 break;
00259             }
00260             case EMB_POLYGON: {
00261                 embArray_free(g.object.polygon.pointList);
00262                 break;
00263             }
00264             case EMB_POLYLINE: {
00265                 embArray_free(g.object.polyline.pointList);
00266                 break;
00267             }
00268             default:
00269                 break;
00270             }
00271         }
00272         free(a->geometry);
00273     break;
00274     }
00275 free(a);
00276 }
00277

```

4.3 src/compress.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "embroidery_internal.h"

```

Functions

- int `hus_compress` (char *data, int length, char *output, int *output_length)
- void `huffman_build_table` (`huffman` *h)
- int * `huffman_lookup` (`huffman` h, int byte_lookup)
- void `compress_init` ()
- int `compress_get_bits` (`compress` *c, int length)
- int `compress_pop` (`compress` *c, int bit_count)
- int `compress_peek` (`compress` *c, int bit_count)
- int `compress_read_variable_length` (`compress` *c)
- void `compress_load_character_length_huffman` (`compress` *c)
- void `compress_load_character_huffman` (`compress` *c)
- void `compress_load_distance_huffman` (`compress` *c)
- void `compress_load_block` (`compress` *c)
- int `compress_get_token` (`compress` *c)
- int `compress_get_position` (`compress` *c)
- int `hus_decompress` (char *data, int length, char *output, int *output_length)

Variables

- int `huffman_lookup_data` [2]

4.3.1 Function Documentation

4.3.1.1 compress_get_bits()

```
int compress_get_bits (
    compress * c,
    int length )
```

Definition at line 106 of file [compress.c](#).

4.3.1.2 compress_get_position()

```
int compress_get_position (
    compress * c )
```

Definition at line 251 of file [compress.c](#).

4.3.1.3 compress_get_token()

```
int compress_get_token (
    compress * c )
```

Definition at line 239 of file [compress.c](#).

4.3.1.4 compress_init()

```
void compress_init ( )
```

Definition at line 101 of file [compress.c](#).

4.3.1.5 compress_load_block()

```
void compress_load_block (
    compress * c )
```

Definition at line 231 of file [compress.c](#).

4.3.1.6 compress_load_character_huffman()

```
void compress_load_character_huffman (
    compress * c )
```

Definition at line 184 of file [compress.c](#).

4.3.1.7 compress_load_character_length_huffman()

```
void compress_load_character_length_huffman (
    compress * c )
```

Definition at line 162 of file [compress.c](#).

4.3.1.8 compress_load_distance_huffman()

```
void compress_load_distance_huffman (
    compress * c )
```

Definition at line 215 of file [compress.c](#).

4.3.1.9 compress_peek()

```
int compress_peek (
    compress * c,
    int bit_count )
```

Definition at line 138 of file [compress.c](#).

4.3.1.10 compress_pop()

```
int compress_pop (
    compress * c,
    int bit_count )
```

Definition at line 131 of file [compress.c](#).

4.3.1.11 compress_read_variable_length()

```
int compress_read_variable_length (
    compress * c )
```

Definition at line 143 of file [compress.c](#).

4.3.1.12 huffman_build_table()

```
void huffman_build_table (
    huffman * h )
```

Definition at line 60 of file [compress.c](#).

4.3.1.13 huffman_lookup()

```
int * huffman_lookup (
    huffman h,
    int byte_lookup )
```

Definition at line 87 of file [compress.c](#).

4.3.1.14 hus_compress()

```
int hus_compress (
    char * data,
    int length,
    char * output,
    int * output_length )
```

Definition at line 44 of file [compress.c](#).

4.3.1.15 hus_decompress()

```
int hus_decompress (
    char * data,
    int length,
    char * output,
    int * output_length )
```

Definition at line 264 of file [compress.c](#).

4.3.2 Variable Documentation

4.3.2.1 huffman_lookup_data

```
int huffman_lookup_data[2]
```

Definition at line 85 of file [compress.c](#).

4.4 compress.c

[Go to the documentation of this file.](#)

```
00001 /*
00002 * This file is part of libembroidery.
00003 *
00004 * Copyright 2018-2022 The Embroidermodder Team
00005 * Licensed under the terms of the zlib license.
00006 *
00007 * This file contains all the read and write functions for the
00008 * library.
00009 *
00010 ****
00011 *
00012 * Thanks to Jason Weiler for describing the binary formats of the HUS and
00013 * VIP formats at:
00014 *
00015 * http://www.jasonweiler.com/HUSandVIPFileInfo.html
00016 *
00017 * Further thanks to github user tatarize for solving the mystery of the
00018 * compression in:
00019 *
00020 * https://github.com/EmbroidePy/pyembroidery
00021 *
00022 * with a description of that work here:
00023 *
00024 * https://stackoverflow.com/questions/7852670/greenleaf-archive-library
00025 *
00026 * This is based on their work.
00027 ****
00028 */
00029
00030 #include <stdio.h>
00031 #include <stdlib.h>
00032 #include <string.h>
00033
00034 #include "embroidery_internal.h"
00035
00036 /* This is a work in progress.
00037 * -----
00038 */
00039
00040 /* This avoids the now unnecessary compression by placing a
00041 * minimal header of 6 bytes and using only literals in the
00042 * huffman compressed part (see the sources above).
00043 */
00044 int hus_compress(char *data, int length, char *output, int *output_length)
00045 {
00046     unsigned char *a = (unsigned char*)output;
00047     a[0] = length%256;
00048     a[1] = (length/256)%256;
00049     a[2] = 0x02;
00050     a[3] = 0xA0;
00051     a[4] = 0x01;
00052     a[5] = 0xFE;
00053     memcpy(output+6, data, length);
00054     *output_length = length+6;
00055     return 0;
00056 }
00057
00058 /* These next 2 functions represent the Huffman class in tartarize's code.
00059 */
00060 void huffman_build_table(huffman *h)
00061 {
```

```

00062     int bit_length, i, max_length, size;
00063     max_length = 0;
00064     size = 1 << h->table_width;
00065     for (i = 0; i < h->table_width; i++) {
00066         if (h->lengths[i] > max_length) {
00067             max_length = h->lengths[i];
00068         }
00069     }
00070     for (bit_length=1; bit_length<=h->table_width; bit_length++) {
00071         int j;
00072         size /= 2;
00073         for (j=0; j < h->nlengths; j++) {
00074             if (h->lengths[j] == bit_length) {
00075                 int k;
00076                 for (k=0; k<size; k++) {
00077                     h->table[h->ntable+k] = j;
00078                     h->ntable++;
00079                 }
00080             }
00081         }
00082     }
00083 }
00084
00085 int huffman_lookup_data[2];
00086
00087 int *huffman_lookup(huffman h, int byte_lookup)
00088 {
00089     int *out = huffman_lookup_data;
00090     if (h.table_width == 0) {
00091         out[0] = h.default_value;
00092         out[1] = 0;
00093         return out;
00094     }
00095     out[0] = h.table[byte_lookup >> (16-h.table_width)];
00096     out[1] = h.lengths[out[0]];
00097     return out;
00098 }
00099
00100 /* These functions represent the EmbCompress class. */
00101 void compress_init()
00102 {
00103
00104 }
00105
00106 int compress_get_bits(compress *c, int length)
00107 {
00108     int i, end_pos_in_bits, start_pos_in_bytes,
00109         end_pos_in_bytes, value, mask_sample_bits,
00110         unused_bits, original;
00111
00112     end_pos_in_bits = c->bit_position + length - 1;
00113     start_pos_in_bytes = c->bit_position / 8;
00114     end_pos_in_bytes = end_pos_in_bits / 8;
00115     value = 0;
00116
00117     for (i=start_pos_in_bytes; i < end_pos_in_bytes+1; i++) {
00118         value <<= 8;
00119         if (i > c->input_length) {
00120             break;
00121         }
00122         value |= (c->input_data[i]) & 0xFF;
00123     }
00124
00125     unused_bits = (7 - end_pos_in_bits) % 8;
00126     mask_sample_bits = (1<<length) - 1;
00127     original = (value >> unused_bits) & mask_sample_bits;
00128     return original;
00129 }
00130
00131 int compress_pop(compress *c, int bit_count)
00132 {
00133     int value = compress_get_bits(c, bit_count);
00134     c->bit_position += bit_count;
00135     return value;
00136 }
00137
00138 int compress_peek(compress *c, int bit_count)
00139 {
00140     return compress_get_bits(c, bit_count);
00141 }
00142
00143 int compress_read_variable_length(compress *c)
00144 {
00145     int q, m, s;
00146     m = compress_pop(c, 3);
00147     if (m!=7) {
00148         return m;

```

```

00149     }
00150     for (q=0; q<13; q++) {
00151         s = compress_pop(c, 1);
00152         if (s) {
00153             m++;
00154         }
00155         else {
00156             break;
00157         }
00158     }
00159     return m;
00160 }
00161
00162 void compress_load_character_length_huffman(compress *c)
00163 {
00164     int count;
00165     count = compress_pop(c, 5);
00166     if (count == 0) {
00167         c->character_length_huffman.default_value = compress_pop(c, 5);
00168     }
00169     else {
00170         int i;
00171         for (i = 0; i < count; i++) {
00172             c->character_length_huffman.lengths[i] = 0;
00173         }
00174         for (i = 0; i < count; i++) {
00175             if (i==3) {
00176                 i += compress_pop(c, 2);
00177             }
00178             c->character_length_huffman.lengths[i] = compress_read_variable_length(c);
00179         }
00180     }
00181     huffman_build_table(&(c->character_length_huffman));
00182 }
00183
00184 void compress_load_character_huffman(compress *c)
00185 {
00186     int count;
00187     count = compress_pop(c, 9);
00188     if (count == 0) {
00189         c->character_huffman.default_value = compress_pop(c, 9);
00190     }
00191     else {
00192         int i = 0;
00193         while (i < count) {
00194             int *h = huffman_lookup(c->character_length_huffman,
00195                                     compress_peek(c, 16));
00196             c->bit_position += h[1];
00197             if (h[0]==0) {
00198                 i += h[0];
00199             }
00200             else if (h[0]==1) {
00201                 i += 3 + compress_pop(c, 4);
00202             }
00203             else if (h[0]==2) {
00204                 i += 20 + compress_pop(c, 9);
00205             }
00206             else {
00207                 c->character_huffman.lengths[i] = h[0] - 2;
00208                 i++;
00209             }
00210         }
00211     }
00212     huffman_build_table(&(c->character_huffman));
00213 }
00214
00215 void compress_load_distance_huffman(compress *c)
00216 {
00217     int count;
00218     count = compress_pop(c, 5);
00219     if (count == 0) {
00220         c->distance_huffman.default_value = compress_pop(c, 5);
00221     }
00222     else {
00223         int i;
00224         for (i = 0; i < count; i++) {
00225             c->distance_huffman.lengths[i] = compress_read_variable_length(c);
00226         }
00227     }
00228     huffman_build_table(&(c->distance_huffman));
00229 }
00230
00231 void compress_load_block(compress *c)
00232 {
00233     c->block_elements = compress_pop(c, 16);
00234     compress_load_character_length_huffman(c);
00235     compress_load_character_huffman(c);

```

```

00236     compress_load_distance_huffman(c);
00237 }
00238
00239 int compress_get_token(compress *c)
00240 {
00241     int *h;
00242     if (c->block_elements <= 0) {
00243         compress_load_block(c);
00244     }
00245     c->block_elements--;
00246     h = huffman_lookup(c->character_huffman, compress_peek(c, 16));
00247     c->bit_position += h[1];
00248     return h[0];
00249 }
00250
00251 int compress_get_position(compress *c)
00252 {
00253     int *h, v;
00254     h = huffman_lookup(c->distance_huffman, compress_peek(c, 16));
00255     c->bit_position += h[1];
00256     if (h[0] == 0) {
00257         return 0;
00258     }
00259     v = h[0] - 1;
00260     v = (1<<v) + compress_pop(c, v);
00261     return v;
00262 }
00263
00264 int hus_decompress(char *data, int length, char *output, int *output_length)
00265 {
00266     int character, i, j;
00267     compress *c = (compress*)malloc(sizeof(compress));
00268     c->bit_position = 0;
00269     c->input_data = data;
00270     c->input_length = length;
00271     c->bits_total = length*8;
00272     i = 0;
00273     while (c->bits_total > c->bit_position && i < *output_length) {
00274         /* process token */
00275         character = 0; /* fix this */
00276         if (character < 0x100) {
00277             output[i] = (char)character;
00278             i++;
00279         }
00280         else if (character == 510) {
00281             break;
00282         }
00283         else {
00284             length = character - 253;
00285             /* not sure about i here */
00286             c->bit_position = i - compress_get_position(c) - 1;
00287             for (j=c->bit_position; j < c->bit_position+length; j++) {
00288                 output[i] = output[j];
00289                 i++;
00290             }
00291         }
00292     }
00293     free(c);
00294     return 0;
00295 }

```

4.5 src/embroidery.h File Reference

Data Structures

- struct EmbColor_
- struct EmbVector_
- struct EmbImage_
- struct EmbBlock_
- struct EmbAlignedDim_
- struct EmbAngularDim_
- struct EmbArcLengthDim_
- struct EmbDiameterDim_
- struct EmbLeaderDim_

- struct EmbLinearDim_
- struct EmbOrdinateDim_
- struct EmbRadiusDim_
- struct EmbInfiniteLine_
- struct EmbRay_
- struct EmbTextMulti_
- struct EmbTextSingle_
- struct EmbTime_
- struct EmbPoint_
- struct EmbLine_
- struct EmbPath_
- struct EmbStitch_
- struct EmbThread_
- struct thread_color_
- struct EmbArc_
- struct EmbRect_
- struct EmbCircle_
- struct EmbSatinOutline_
- struct EmbEllipse_
- struct EmbBezier_
- struct EmbSpline_
- struct LSYSTEM
- struct EmbGeometry_
- struct EmbArray_
- struct EmbLayer_
- struct EmbPattern_
- struct EmbFormatList_

Macros

- #define LIBEMBROIDERY_EMBEDDED_VERSION 0
- #define NORMAL 0 /* stitch to (x, y) */
- #define JUMP 1 /* move to (x, y) */
- #define TRIM 2 /* trim + move to (x, y) */
- #define STOP 4 /* pause machine for thread change */
- #define SEQUIN 8 /* sequin */
- #define END 16 /* end of program */
- #define EMB_FORMAT_100 0
- #define EMB_FORMAT_10O 1
- #define EMB_FORMAT_ART 2
- #define EMB_FORMAT_BMC 3
- #define EMB_FORMAT_BRO 4
- #define EMB_FORMAT_CND 5
- #define EMB_FORMAT_COL 6
- #define EMB_FORMAT_CSD 7
- #define EMB_FORMAT_CSV 8
- #define EMB_FORMAT_DAT 9
- #define EMB_FORMAT_DEM 10
- #define EMB_FORMAT_DSB 11
- #define EMB_FORMAT_DST 12
- #define EMB_FORMAT_DSZ 13
- #define EMB_FORMAT_DXF 14
- #define EMB_FORMAT_EDR 15
- #define EMB_FORMAT_EMD 16

- #define EMB_FORMAT_EXP 17
- #define EMB_FORMAT_EXY 18
- #define EMB_FORMAT_EYS 19
- #define EMB_FORMAT_FXY 20
- #define EMB_FORMAT_GC 21
- #define EMB_FORMAT_GNC 22
- #define EMB_FORMAT_GT 23
- #define EMB_FORMAT_HUS 24
- #define EMB_FORMAT_INB 25
- #define EMB_FORMAT_INF 26
- #define EMB_FORMAT_JEF 27
- #define EMB_FORMAT_KSM 28
- #define EMB_FORMAT_MAX 29
- #define EMB_FORMAT_MIT 30
- #define EMB_FORMAT_NEW 31
- #define EMB_FORMAT_OFM 32
- #define EMB_FORMAT_PCD 33
- #define EMB_FORMAT_PCM 34
- #define EMB_FORMAT_PCQ 35
- #define EMB_FORMAT_PCS 36
- #define EMB_FORMAT_PEC 37
- #define EMB_FORMAT_PEL 38
- #define EMB_FORMAT_PEM 39
- #define EMB_FORMAT_PES 40
- #define EMB_FORMAT_PHB 41
- #define EMB_FORMAT_PHC 42
- #define EMB_FORMAT_PLT 43
- #define EMB_FORMAT_RGB 44
- #define EMB_FORMAT_SEW 45
- #define EMB_FORMAT_SHV 46
- #define EMB_FORMAT_SST 47
- #define EMB_FORMAT_STX 48
- #define EMB_FORMAT_SVG 49
- #define EMB_FORMAT_T01 50
- #define EMB_FORMAT_T09 51
- #define EMB_FORMAT_TAP 52
- #define EMB_FORMAT_THR 53
- #define EMB_FORMAT_TXT 54
- #define EMB_FORMAT_U00 55
- #define EMB_FORMAT_U01 56
- #define EMB_FORMAT_VIP 57
- #define EMB_FORMAT_VP3 58
- #define EMB_FORMAT_XXX 59
- #define EMB_FORMAT_ZSK 60
- #define Arc_Polyester 0
- #define Arc_Rayon 1
- #define CoatsAndClark_Rayon 2
- #define Exquisite_Polyester 3
- #define Fufu_Polyester 4
- #define Fufu_Rayon 5
- #define Hemingworth_Polyester 6
- #define Isacord_Polyester 7
- #define Isafil_Rayon 8
- #define Marathon_Polyester 9
- #define Marathon_Rayon 10

- #define Madeira_Polyester 11
- #define Madeira_Rayon 12
- #define Metro_Polyester 13
- #define Pantone 14
- #define RobisonAnton_Polyester 15
- #define RobisonAnton_Rayon 16
- #define Sigma_Polyester 17
- #define Sulky_Rayon 18
- #define ThreadArt_Rayon 19
- #define ThreadArt_Polyester 20
- #define ThreaDelight_Polyester 21
- #define Z102_Isacord_Polyester 22
- #define SVG_Colors 23
- #define hus_thread 24
- #define jef_thread 25
- #define pcm_thread 26
- #define pec_thread 27
- #define shv_thread 28
- #define dxf_color 29
- #define EMB_ARRAY 0
- #define EMB_ARC 1
- #define EMB_CIRCLE 2
- #define EMB_DIM_DIAMETER 3
- #define EMB_DIM_LEADER 4
- #define EMB_ELLIPSE 5
- #define EMB_FLAG 6
- #define EMB_LINE 7
- #define EMB_IMAGE 8
- #define EMB_PATH 9
- #define EMB_POINT 10
- #define EMB_POLYGON 11
- #define EMB_POLYLINE 12
- #define EMB_RECT 13
- #define EMB_SPLINE 14
- #define EMB_STITCH 15
- #define EMB_TEXT_SINGLE 16
- #define EMB_TEXT_MULTI 17
- #define EMB_VECTOR 18
- #define EMB_THREAD 19
- #define EMBFORMAT_UNSUPPORTED 0
- #define EMBFORMAT_STITCHONLY 1
- #define EMBFORMAT_OBJECTONLY 2
- #define EMBFORMAT_STCHANDOBJ 3 /* binary operation: 1+2=3 */
- #define numberOfFormats 61
- #define CHUNK_SIZE 128
- #define EMB_MAX_LAYERS 10
- #define MAX_THREADS 256
- #define EMBFORMAT_MAXEXT 3
- #define EMBFORMAT_MAXDESC 50
- #define MAX_STITCHES 1000000
- #define EMB_PUBLIC

Typedefs

- `typedef float EmbReal`
- `typedef struct EmbColor_ EmbColor`
- `typedef struct EmbVector_ EmbVector`
- `typedef struct EmbArray_ EmbArray`
- `typedef struct EmbImage_ EmbImage`
- `typedef struct EmbBlock_ EmbBlock`
- `typedef struct EmbAlignedDim_ EmbAlignedDim`
- `typedef struct EmbAngularDim_ EmbAngularDim`
- `typedef struct EmbArcLengthDim_ EmbArcLengthDim`
- `typedef struct EmbDiameterDim_ EmbDiameterDim`
- `typedef struct EmbLeaderDim_ EmbLeaderDim`
- `typedef struct EmbLinearDim_ EmbLinearDim`
- `typedef struct EmbOrdinateDim_ EmbOrdinateDim`
- `typedef struct EmbRadiusDim_ EmbRadiusDim`
- `typedef struct EmbInfiniteLine_ EmbInfiniteLine`
- `typedef struct EmbRay_ EmbRay`
- `typedef struct EmbTextMulti_ EmbTextMulti`
- `typedef struct EmbTextSingle_ EmbTextSingle`
- `typedef struct EmbTime_ EmbTime`
- `typedef struct EmbPoint_ EmbPoint`
- `typedef struct EmbLine_ EmbLine`
- `typedef struct EmbPath_ EmbPath`
- `typedef struct EmbStitch_ EmbStitch`
- `typedef struct EmbThread_ EmbThread`
- `typedef struct thread_color_ thread_color`
- `typedef struct EmbArc_ EmbArc`
- `typedef struct EmbRect_ EmbRect`
- `typedef struct EmbCircle_ EmbCircle`
- `typedef EmbPath EmbPolygon`
- `typedef EmbPath EmbPolyline`
- `typedef int EmbFlag`
- `typedef struct EmbSatinOutline_ EmbSatinOutline`
- `typedef struct EmbEllipse_ EmbEllipse`
- `typedef struct EmbBezier_ EmbBezier`
- `typedef struct EmbSpline_ EmbSpline`
- `typedef struct LSYSTEM L_system`
- `typedef struct EmbGeometry_ EmbGeometry`
- `typedef struct EmbLayer_ EmbLayer`
- `typedef struct EmbPattern_ EmbPattern`
- `typedef struct EmbFormatList_ EmbFormatList`

Functions

- `EMB_PUBLIC int lindenmayer_system (L_system L, char *state, int iteration, int complete)`
- `EMB_PUBLIC int hilbert_curve (EmbPattern *pattern, int iterations)`
- `EMB_PUBLIC int emb_identify_format (const char *ending)`
- `EMB_PUBLIC void testMain (int level)`
- `EMB_PUBLIC int convert (const char *inf, const char *outf)`
- `EMB_PUBLIC EmbColor embColor_make (unsigned char r, unsigned char g, unsigned char b)`
- `EMB_PUBLIC EmbColor * embColor_create (unsigned char r, unsigned char g, unsigned char b)`
- `EMB_PUBLIC EmbColor embColor_fromHexStr (char *val)`
- `EMB_PUBLIC int embColor_distance (EmbColor a, EmbColor b)`

- EMB_PUBLIC EmbArray * embArray_create (int type)
- EMB_PUBLIC int embArray_resize (EmbArray *g)
- EMB_PUBLIC void embArray_copy (EmbArray *dst, EmbArray *src)
- EMB_PUBLIC int embArray_addArc (EmbArray *g, EmbArc arc)
- EMB_PUBLIC int embArray_addCircle (EmbArray *g, EmbCircle circle)
- EMB_PUBLIC int embArray_addEllipse (EmbArray *g, EmbEllipse ellipse)
- EMB_PUBLIC int embArray_addFlag (EmbArray *g, int flag)
- EMB_PUBLIC int embArray_addLine (EmbArray *g, EmbLine line)
- EMB_PUBLIC int embArray_addRect (EmbArray *g, EmbRect rect)
- EMB_PUBLIC int embArray_addPath (EmbArray *g, EmbPath p)
- EMB_PUBLIC int embArray_addPoint (EmbArray *g, EmbPoint p)
- EMB_PUBLIC int embArray_addPolygon (EmbArray *g, EmbPolygon p)
- EMB_PUBLIC int embArray_addPolyline (EmbArray *g, EmbPolyline p)
- EMB_PUBLIC int embArray_addStitch (EmbArray *g, EmbStitch st)
- EMB_PUBLIC int embArray_addThread (EmbArray *g, EmbThread p)
- EMB_PUBLIC int embArray_addVector (EmbArray *g, EmbVector)
- EMB_PUBLIC void embArray_free (EmbArray *p)
- EMB_PUBLIC EmbLine embLine_make (EmbReal x1, EmbReal y1, EmbReal x2, EmbReal y2)
- EMB_PUBLIC void embLine_normalVector (EmbLine line, EmbVector *result, int clockwise)
- EMB_PUBLIC EmbVector embLine_intersectionPoint (EmbLine line1, EmbLine line2)
- EMB_PUBLIC int embThread_findNearestColor (EmbColor color, EmbColor *colors, int n_colors)
- EMB_PUBLIC int embThread_findNearestThread (EmbColor color, EmbThread *threads, int n_threads)
- EMB_PUBLIC EmbThread embThread_getRandom (void)
- EMB_PUBLIC void embVector_normalize (EmbVector vector, EmbVector *result)
- EMB_PUBLIC void embVector_multiply (EmbVector vector, EmbReal magnitude, EmbVector *result)
- EMB_PUBLIC EmbVector embVector_add (EmbVector v1, EmbVector v2)
- EMB_PUBLIC EmbVector embVector_average (EmbVector v1, EmbVector v2)
- EMB_PUBLIC EmbVector embVector_subtract (EmbVector v1, EmbVector v2)
- EMB_PUBLIC EmbReal embVector_dot (EmbVector v1, EmbVector v2)
- EMB_PUBLIC EmbReal embVector_cross (EmbVector v1, EmbVector v2)
- EMB_PUBLIC void embVector_transpose_product (EmbVector v1, EmbVector v2, EmbVector *result)
- EMB_PUBLIC EmbReal embVector_length (EmbVector vector)
- EMB_PUBLIC EmbReal embVector_relativeX (EmbVector a1, EmbVector a2, EmbVector a3)
- EMB_PUBLIC EmbReal embVector_relativeY (EmbVector a1, EmbVector a2, EmbVector a3)
- EMB_PUBLIC EmbReal embVector_angle (EmbVector v)
- EMB_PUBLIC EmbReal embVector_distance (EmbVector a, EmbVector b)
- EMB_PUBLIC EmbVector embVector_unit (EmbReal angle)
- EMB_PUBLIC EmbArc embArc_init (void)
- EMB_PUBLIC char embArc_clockwise (EmbArc arc)
- EMB_PUBLIC void getArcCenter (EmbArc arc, EmbVector *arcCenter)
- EMB_PUBLIC char getArcDataFromBulge (EmbReal bulge, EmbArc *arc, EmbReal *arcCenterX, EmbReal *arcCenterY, EmbReal *radius, EmbReal *diameter, EmbReal *chord, EmbReal *chordMidX, EmbReal *chordMidY, EmbReal *sagitta, EmbReal *apothem, EmbReal *incAngleInDegrees, char *clockwise)
- EMB_PUBLIC EmbCircle embCircle_init (void)
- EMB_PUBLIC int getCircleCircleIntersections (EmbCircle c0, EmbCircle c1, EmbVector *v0, EmbVector *v1)
- EMB_PUBLIC int getCircleTangentPoints (EmbCircle c, EmbVector p, EmbVector *v0, EmbVector *v1)
- EMB_PUBLIC EmbEllipse embEllipse_init (void)
- EMB_PUBLIC EmbEllipse embEllipse_make (EmbReal cx, EmbReal cy, EmbReal rx, EmbReal ry)
- EMB_PUBLIC EmbReal embEllipse_diameterX (EmbEllipse ellipse)
- EMB_PUBLIC EmbReal embEllipse_diameterY (EmbEllipse ellipse)
- EMB_PUBLIC EmbReal embEllipse_width (EmbEllipse ellipse)
- EMB_PUBLIC EmbReal embEllipse_height (EmbEllipse ellipse)
- EMB_PUBLIC EmbReal embEllipse_area (EmbEllipse ellipse)
- EMB_PUBLIC EmbReal embEllipse_perimeter (EmbEllipse ellipse)
- EMB_PUBLIC EmblImage emblImage_create (int, int)

- EMB_PUBLIC void `emblImage_read` (`EmblImage` *image, char *fname)
- EMB_PUBLIC int `emblImage_write` (`EmblImage` *image, char *fname)
- EMB_PUBLIC void `emblImage_free` (`EmblImage` *image)
- EMB_PUBLIC `EmbRect` `embRect_init` (void)
- EMB_PUBLIC `EmbReal` `embRect_area` (`EmbRect`)
- EMB_PUBLIC int `threadColor` (const char *, int brand)
- EMB_PUBLIC int `threadColorNum` (unsigned int color, int brand)
- EMB_PUBLIC const char * `threadColorName` (unsigned int color, int brand)
- EMB_PUBLIC void `embTime_initNow` (`EmbTime` *t)
- EMB_PUBLIC `EmbTime` `embTime_time` (`EmbTime` *t)
- EMB_PUBLIC void `embSatinOutline_generateSatinOutline` (`EmbArray` *lines, `EmbReal` thickness, `EmbSatinOutline` *result)
- EMB_PUBLIC `EmbArray` * `embSatinOutline_renderStitches` (`EmbSatinOutline` *result, `EmbReal` density)
- EMB_PUBLIC `EmbGeometry` * `embGeometry_init` (int type_in)
- EMB_PUBLIC void `embGeometry_free` (`EmbGeometry` *obj)
- EMB_PUBLIC void `embGeometry_move` (`EmbGeometry` *obj, `EmbVector` delta)
- EMB_PUBLIC `EmbRect` `embGeometry_boundingRect` (`EmbGeometry` *obj)
- EMB_PUBLIC void `embGeometry_vulcanize` (`EmbGeometry` *obj)
- EMB_PUBLIC `EmbPattern` * `embPattern_create` (void)
- EMB_PUBLIC void `embPattern_hideStitchesOverLength` (`EmbPattern` *p, int length)
- EMB_PUBLIC void `embPattern_fixColorCount` (`EmbPattern` *p)
- EMB_PUBLIC int `embPattern_addThread` (`EmbPattern` *p, `EmbThread` thread)
- EMB_PUBLIC void `embPattern_addStitchAbs` (`EmbPattern` *p, `EmbReal` x, `EmbReal` y, int flags, int isAuto← ColorIndex)
- EMB_PUBLIC void `embPattern_addStitchRel` (`EmbPattern` *p, `EmbReal` dx, `EmbReal` dy, int flags, int is← AutoColorIndex)
- EMB_PUBLIC void `embPattern_changeColor` (`EmbPattern` *p, int index)
- EMB_PUBLIC void `embPattern_free` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_scale` (`EmbPattern` *p, `EmbReal` scale)
- EMB_PUBLIC `EmbReal` `embPattern_totalStitchLength` (`EmbPattern` *pattern)
- EMB_PUBLIC `EmbReal` `embPattern_minimumStitchLength` (`EmbPattern` *pattern)
- EMB_PUBLIC `EmbReal` `embPattern_maximumStitchLength` (`EmbPattern` *pattern)
- EMB_PUBLIC void `embPattern_lengthHistogram` (`EmbPattern` *pattern, int *bin, int NUMBINS)
- EMB_PUBLIC int `embPattern_realStitches` (`EmbPattern` *pattern)
- EMB_PUBLIC int `embPattern_jumpStitches` (`EmbPattern` *pattern)
- EMB_PUBLIC int `embPattern_trimStitches` (`EmbPattern` *pattern)
- EMB_PUBLIC `EmbRect` `embPattern_calcBoundingBox` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_flipHorizontal` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_flipVertical` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_flip` (`EmbPattern` *p, int horz, int vert)
- EMB_PUBLIC void `embPattern_combineJumpStitches` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_correctForMaxStitchLength` (`EmbPattern` *p, `EmbReal` maxStitchLength, `EmbReal` maxJumpLength)
- EMB_PUBLIC void `embPattern_center` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_loadExternalColorFile` (`EmbPattern` *p, const char *fileName)
- EMB_PUBLIC void `embPattern_convertGeometry` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_designDetails` (`EmbPattern` *p)
- EMB_PUBLIC `EmbPattern` * `embPattern_combine` (`EmbPattern` *p1, `EmbPattern` *p2)
- EMB_PUBLIC int `embPattern_color_count` (`EmbPattern` *pattern, `EmbColor` startColor)
- EMB_PUBLIC void `embPattern_end` (`EmbPattern` *p)
- EMB_PUBLIC void `embPattern_crossstitch` (`EmbPattern` *pattern, `EmblImage` *, int threshhold)
- EMB_PUBLIC void `embPattern_horizontal_fill` (`EmbPattern` *pattern, `EmblImage` *, int threshhold)
- EMB_PUBLIC int `embPattern_render` (`EmbPattern` *pattern, char *fname)
- EMB_PUBLIC int `embPattern_simulate` (`EmbPattern` *pattern, char *fname)
- EMB_PUBLIC void `embPattern_addCircleAbs` (`EmbPattern` *p, `EmbCircle` obj)

- EMB_PUBLIC void `embPattern_addEllipseAbs` (EmbPattern *p, EmbEllipse obj)
- EMB_PUBLIC void `embPattern_addLineAbs` (EmbPattern *p, EmbLine obj)
- EMB_PUBLIC void `embPattern_addPathAbs` (EmbPattern *p, EmbPath obj)
- EMB_PUBLIC void `embPattern_addPointAbs` (EmbPattern *p, EmbPoint obj)
- EMB_PUBLIC void `embPattern_addPolygonAbs` (EmbPattern *p, EmbPolygon obj)
- EMB_PUBLIC void `embPattern_addPolylineAbs` (EmbPattern *p, EmbPolyline obj)
- EMB_PUBLIC void `embPattern_addRectAbs` (EmbPattern *p, EmbRect obj)
- EMB_PUBLIC void `embPattern_copyStitchListToPolylines` (EmbPattern *pattern)
- EMB_PUBLIC void `embPattern_copyPolylinesToStitchList` (EmbPattern *pattern)
- EMB_PUBLIC void `embPattern_moveStitchListToPolylines` (EmbPattern *pattern)
- EMB_PUBLIC void `embPattern_movePolylinesToStitchList` (EmbPattern *pattern)
- EMB_PUBLIC char `embPattern_read` (EmbPattern *pattern, const char *fileName, int format)
- EMB_PUBLIC char `embPattern_write` (EmbPattern *pattern, const char *fileName, int format)
- EMB_PUBLIC char `embPattern_readAuto` (EmbPattern *pattern, const char *fileName)
- EMB_PUBLIC char `embPattern_writeAuto` (EmbPattern *pattern, const char *fileName)
- EMB_PUBLIC void `report` (int result, char *label)
- EMB_PUBLIC int `full_test_matrix` (char *fname)
- EMB_PUBLIC int `emb_round` (EmbReal x)
- EMB_PUBLIC EmbReal `radians` (EmbReal degree)
- EMB_PUBLIC EmbReal `degrees` (EmbReal radian)

Variables

- EmbFormatList `formatTable` [numberOfFormats]
- const int `pecThreadCount`
- const int `shvThreadCount`
- const EmbReal `embConstantPi`
- const EmbThread `husThreads` []
- const EmbThread `jefThreads` []
- const EmbThread `shvThreads` []
- const EmbThread `pcmThreads` []
- const EmbThread `pecThreads` []
- const unsigned char `_dxfColorTable` [][]3
- EmbThread `black_thread`
- const unsigned char `vipDecodingTable` []
- int `emb_error`
- int `emb_verbose`

4.5.1 Macro Definition Documentation

4.5.1.1 Arc_Polyester

```
#define Arc_Polyester 0
```

Definition at line 87 of file `embroidery.h`.

4.5.1.2 Arc_Rayon

```
#define Arc_Rayon 1
```

Definition at line 88 of file [embroidery.h](#).

4.5.1.3 CHUNK_SIZE

```
#define CHUNK_SIZE 128
```

Definition at line 146 of file [embroidery.h](#).

4.5.1.4 CoatsAndClark_Rayon

```
#define CoatsAndClark_Rayon 2
```

Definition at line 89 of file [embroidery.h](#).

4.5.1.5 dxf_color

```
#define dxf_color 29
```

Definition at line 116 of file [embroidery.h](#).

4.5.1.6 EMB_ARC

```
#define EMB_ARC 1
```

Definition at line 119 of file [embroidery.h](#).

4.5.1.7 EMB_ARRAY

```
#define EMB_ARRAY 0
```

Definition at line 118 of file [embroidery.h](#).

4.5.1.8 EMB_CIRCLE

```
#define EMB_CIRCLE 2
```

Definition at line 120 of file [embroidery.h](#).

4.5.1.9 EMB_DIM_DIAMETER

```
#define EMB_DIM_DIAMETER 3
```

Definition at line 121 of file [embroidery.h](#).

4.5.1.10 EMB_DIM_LEADER

```
#define EMB_DIM_LEADER 4
```

Definition at line 122 of file [embroidery.h](#).

4.5.1.11 EMB_ELLIPSE

```
#define EMB_ELLIPSE 5
```

Definition at line 123 of file [embroidery.h](#).

4.5.1.12 EMB_FLAG

```
#define EMB_FLAG 6
```

Definition at line 124 of file [embroidery.h](#).

4.5.1.13 EMB_FORMAT_100

```
#define EMB_FORMAT_100 0
```

Definition at line 24 of file [embroidery.h](#).

4.5.1.14 EMB_FORMAT_100

```
#define EMB_FORMAT_100 1
```

Definition at line [25](#) of file [embroidery.h](#).

4.5.1.15 EMB_FORMAT_ART

```
#define EMB_FORMAT_ART 2
```

Definition at line [26](#) of file [embroidery.h](#).

4.5.1.16 EMB_FORMAT_BMC

```
#define EMB_FORMAT_BMC 3
```

Definition at line [27](#) of file [embroidery.h](#).

4.5.1.17 EMB_FORMAT_BRO

```
#define EMB_FORMAT_BRO 4
```

Definition at line [28](#) of file [embroidery.h](#).

4.5.1.18 EMB_FORMAT_CND

```
#define EMB_FORMAT_CND 5
```

Definition at line [29](#) of file [embroidery.h](#).

4.5.1.19 EMB_FORMAT_COL

```
#define EMB_FORMAT_COL 6
```

Definition at line [30](#) of file [embroidery.h](#).

4.5.1.20 EMB_FORMAT_CSD

```
#define EMB_FORMAT_CSD 7
```

Definition at line 31 of file [embroidery.h](#).

4.5.1.21 EMB_FORMAT_CSV

```
#define EMB_FORMAT_CSV 8
```

Definition at line 32 of file [embroidery.h](#).

4.5.1.22 EMB_FORMAT_DAT

```
#define EMB_FORMAT_DAT 9
```

Definition at line 33 of file [embroidery.h](#).

4.5.1.23 EMB_FORMAT_DEM

```
#define EMB_FORMAT_DEM 10
```

Definition at line 34 of file [embroidery.h](#).

4.5.1.24 EMB_FORMAT_DSB

```
#define EMB_FORMAT_DSB 11
```

Definition at line 35 of file [embroidery.h](#).

4.5.1.25 EMB_FORMAT_DST

```
#define EMB_FORMAT_DST 12
```

Definition at line 36 of file [embroidery.h](#).

4.5.1.26 EMB_FORMAT_DSZ

```
#define EMB_FORMAT_DSZ 13
```

Definition at line 37 of file [embroidery.h](#).

4.5.1.27 EMB_FORMAT_DXF

```
#define EMB_FORMAT_DXF 14
```

Definition at line 38 of file [embroidery.h](#).

4.5.1.28 EMB_FORMAT_EDR

```
#define EMB_FORMAT_EDR 15
```

Definition at line 39 of file [embroidery.h](#).

4.5.1.29 EMB_FORMAT_EMD

```
#define EMB_FORMAT_EMD 16
```

Definition at line 40 of file [embroidery.h](#).

4.5.1.30 EMB_FORMAT_EXP

```
#define EMB_FORMAT_EXP 17
```

Definition at line 41 of file [embroidery.h](#).

4.5.1.31 EMB_FORMAT_EXY

```
#define EMB_FORMAT_EXY 18
```

Definition at line 42 of file [embroidery.h](#).

4.5.1.32 EMB_FORMAT_EYS

```
#define EMB_FORMAT_EYS 19
```

Definition at line [43](#) of file [embroidery.h](#).

4.5.1.33 EMB_FORMAT_FXY

```
#define EMB_FORMAT_FXY 20
```

Definition at line [44](#) of file [embroidery.h](#).

4.5.1.34 EMB_FORMAT_GC

```
#define EMB_FORMAT_GC 21
```

Definition at line [45](#) of file [embroidery.h](#).

4.5.1.35 EMB_FORMAT_GNC

```
#define EMB_FORMAT_GNC 22
```

Definition at line [46](#) of file [embroidery.h](#).

4.5.1.36 EMB_FORMAT_GT

```
#define EMB_FORMAT_GT 23
```

Definition at line [47](#) of file [embroidery.h](#).

4.5.1.37 EMB_FORMAT_HUS

```
#define EMB_FORMAT_HUS 24
```

Definition at line [48](#) of file [embroidery.h](#).

4.5.1.38 EMB_FORMAT_INB

```
#define EMB_FORMAT_INB 25
```

Definition at line 49 of file [embroidery.h](#).

4.5.1.39 EMB_FORMAT_INF

```
#define EMB_FORMAT_INF 26
```

Definition at line 50 of file [embroidery.h](#).

4.5.1.40 EMB_FORMAT_JEF

```
#define EMB_FORMAT_JEF 27
```

Definition at line 51 of file [embroidery.h](#).

4.5.1.41 EMB_FORMAT_KSM

```
#define EMB_FORMAT_KSM 28
```

Definition at line 52 of file [embroidery.h](#).

4.5.1.42 EMB_FORMAT_MAX

```
#define EMB_FORMAT_MAX 29
```

Definition at line 53 of file [embroidery.h](#).

4.5.1.43 EMB_FORMAT_MIT

```
#define EMB_FORMAT_MIT 30
```

Definition at line 54 of file [embroidery.h](#).

4.5.1.44 EMB_FORMAT_NEW

```
#define EMB_FORMAT_NEW 31
```

Definition at line 55 of file [embroidery.h](#).

4.5.1.45 EMB_FORMAT_OFM

```
#define EMB_FORMAT_OFM 32
```

Definition at line 56 of file [embroidery.h](#).

4.5.1.46 EMB_FORMAT_PCD

```
#define EMB_FORMAT_PCD 33
```

Definition at line 57 of file [embroidery.h](#).

4.5.1.47 EMB_FORMAT_PCM

```
#define EMB_FORMAT_PCM 34
```

Definition at line 58 of file [embroidery.h](#).

4.5.1.48 EMB_FORMAT_PCQ

```
#define EMB_FORMAT_PCQ 35
```

Definition at line 59 of file [embroidery.h](#).

4.5.1.49 EMB_FORMAT_PCS

```
#define EMB_FORMAT_PCS 36
```

Definition at line 60 of file [embroidery.h](#).

4.5.1.50 EMB_FORMAT_PEC

```
#define EMB_FORMAT_PEC 37
```

Definition at line [61](#) of file [embroidery.h](#).

4.5.1.51 EMB_FORMAT_PEL

```
#define EMB_FORMAT_PEL 38
```

Definition at line [62](#) of file [embroidery.h](#).

4.5.1.52 EMB_FORMAT_PEM

```
#define EMB_FORMAT_PEM 39
```

Definition at line [63](#) of file [embroidery.h](#).

4.5.1.53 EMB_FORMAT_PES

```
#define EMB_FORMAT_PES 40
```

Definition at line [64](#) of file [embroidery.h](#).

4.5.1.54 EMB_FORMAT_PHB

```
#define EMB_FORMAT_PHB 41
```

Definition at line [65](#) of file [embroidery.h](#).

4.5.1.55 EMB_FORMAT_PHC

```
#define EMB_FORMAT_PHC 42
```

Definition at line [66](#) of file [embroidery.h](#).

4.5.1.56 EMB_FORMAT_PLT

```
#define EMB_FORMAT_PLT 43
```

Definition at line [67](#) of file [embroidery.h](#).

4.5.1.57 EMB_FORMAT_RGB

```
#define EMB_FORMAT_RGB 44
```

Definition at line [68](#) of file [embroidery.h](#).

4.5.1.58 EMB_FORMAT_SEW

```
#define EMB_FORMAT_SEW 45
```

Definition at line [69](#) of file [embroidery.h](#).

4.5.1.59 EMB_FORMAT_SHV

```
#define EMB_FORMAT_SHV 46
```

Definition at line [70](#) of file [embroidery.h](#).

4.5.1.60 EMB_FORMAT_SST

```
#define EMB_FORMAT_SST 47
```

Definition at line [71](#) of file [embroidery.h](#).

4.5.1.61 EMB_FORMAT_STX

```
#define EMB_FORMAT_STX 48
```

Definition at line [72](#) of file [embroidery.h](#).

4.5.1.62 EMB_FORMAT_SVG

```
#define EMB_FORMAT_SVG 49
```

Definition at line [73](#) of file [embroidery.h](#).

4.5.1.63 EMB_FORMAT_T01

```
#define EMB_FORMAT_T01 50
```

Definition at line [74](#) of file [embroidery.h](#).

4.5.1.64 EMB_FORMAT_T09

```
#define EMB_FORMAT_T09 51
```

Definition at line [75](#) of file [embroidery.h](#).

4.5.1.65 EMB_FORMAT_TAP

```
#define EMB_FORMAT_TAP 52
```

Definition at line [76](#) of file [embroidery.h](#).

4.5.1.66 EMB_FORMAT_THR

```
#define EMB_FORMAT_THR 53
```

Definition at line [77](#) of file [embroidery.h](#).

4.5.1.67 EMB_FORMAT_TXT

```
#define EMB_FORMAT_TXT 54
```

Definition at line [78](#) of file [embroidery.h](#).

4.5.1.68 EMB_FORMAT_U00

```
#define EMB_FORMAT_U00 55
```

Definition at line [79](#) of file [embroidery.h](#).

4.5.1.69 EMB_FORMAT_U01

```
#define EMB_FORMAT_U01 56
```

Definition at line [80](#) of file [embroidery.h](#).

4.5.1.70 EMB_FORMAT_VIP

```
#define EMB_FORMAT_VIP 57
```

Definition at line [81](#) of file [embroidery.h](#).

4.5.1.71 EMB_FORMAT_VP3

```
#define EMB_FORMAT_VP3 58
```

Definition at line [82](#) of file [embroidery.h](#).

4.5.1.72 EMB_FORMAT_XXX

```
#define EMB_FORMAT_XXX 59
```

Definition at line [83](#) of file [embroidery.h](#).

4.5.1.73 EMB_FORMAT_ZSK

```
#define EMB_FORMAT_ZSK 60
```

Definition at line [84](#) of file [embroidery.h](#).

4.5.1.74 EMB_IMAGE

```
#define EMB_IMAGE 8
```

Definition at line 126 of file [embroidery.h](#).

4.5.1.75 EMB_LINE

```
#define EMB_LINE 7
```

Definition at line 125 of file [embroidery.h](#).

4.5.1.76 EMB_MAX_LAYERS

```
#define EMB_MAX_LAYERS 10
```

Definition at line 148 of file [embroidery.h](#).

4.5.1.77 EMB_PATH

```
#define EMB_PATH 9
```

Definition at line 127 of file [embroidery.h](#).

4.5.1.78 EMB_POINT

```
#define EMB_POINT 10
```

Definition at line 128 of file [embroidery.h](#).

4.5.1.79 EMB_POLYGON

```
#define EMB_POLYGON 11
```

Definition at line 129 of file [embroidery.h](#).

4.5.1.80 EMB_POLYLINE

```
#define EMB_POLYLINE 12
```

Definition at line 130 of file [embroidery.h](#).

4.5.1.81 EMB_PUBLIC

```
#define EMB_PUBLIC
```

Definition at line 164 of file [embroidery.h](#).

4.5.1.82 EMB_RECT

```
#define EMB_RECT 13
```

Definition at line 131 of file [embroidery.h](#).

4.5.1.83 EMB_SPLINE

```
#define EMB_SPLINE 14
```

Definition at line 132 of file [embroidery.h](#).

4.5.1.84 EMB_STITCH

```
#define EMB_STITCH 15
```

Definition at line 133 of file [embroidery.h](#).

4.5.1.85 EMB_TEXT_MULTI

```
#define EMB_TEXT_MULTI 17
```

Definition at line 135 of file [embroidery.h](#).

4.5.1.86 EMB_TEXT_SINGLE

```
#define EMB_TEXT_SINGLE 16
```

Definition at line 134 of file [embroidery.h](#).

4.5.1.87 EMB_THREAD

```
#define EMB_THREAD 19
```

Definition at line 137 of file [embroidery.h](#).

4.5.1.88 EMB_VECTOR

```
#define EMB_VECTOR 18
```

Definition at line 136 of file [embroidery.h](#).

4.5.1.89 EMBFORMAT_MAXDESC

```
#define EMBFORMAT_MAXDESC 50
```

Definition at line 152 of file [embroidery.h](#).

4.5.1.90 EMBFORMAT_MAXEXT

```
#define EMBFORMAT_MAXEXT 3
```

Definition at line 150 of file [embroidery.h](#).

4.5.1.91 EMBFORMAT_OBJECTONLY

```
#define EMBFORMAT_OBJECTONLY 2
```

Definition at line 141 of file [embroidery.h](#).

4.5.1.92 EMBFORMAT_STCHANDOBJ

```
#define EMBFORMAT_STCHANDOBJ 3 /* binary operation: 1+2=3 */
```

Definition at line 142 of file [embroidery.h](#).

4.5.1.93 EMBFORMAT_STITCHONLY

```
#define EMBFORMAT_STITCHONLY 1
```

Definition at line 140 of file [embroidery.h](#).

4.5.1.94 EMBFORMAT_UNSUPPORTED

```
#define EMBFORMAT_UNSUPPORTED 0
```

Definition at line 139 of file [embroidery.h](#).

4.5.1.95 END

```
#define END 16 /* end of program */
```

Definition at line 21 of file [embroidery.h](#).

4.5.1.96 Exquisite_Polyester

```
#define Exquisite_Polyester 3
```

Definition at line 90 of file [embroidery.h](#).

4.5.1.97 Fufu_Polyester

```
#define Fufu_Polyester 4
```

Definition at line 91 of file [embroidery.h](#).

4.5.1.98 Fufu_Rayon

```
#define Fufu_Rayon 5
```

Definition at line 92 of file [embroidery.h](#).

4.5.1.99 Hemingworth_Polyester

```
#define Hemingworth_Polyester 6
```

Definition at line 93 of file [embroidery.h](#).

4.5.1.100 hus_thread

```
#define hus_thread 24
```

Definition at line 111 of file [embroidery.h](#).

4.5.1.101 Isacord_Polyester

```
#define Isacord_Polyester 7
```

Definition at line 94 of file [embroidery.h](#).

4.5.1.102 Isafil_Rayon

```
#define Isafil_Rayon 8
```

Definition at line 95 of file [embroidery.h](#).

4.5.1.103 jef_thread

```
#define jef_thread 25
```

Definition at line 112 of file [embroidery.h](#).

4.5.1.104 JUMP

```
#define JUMP 1 /* move to (x, y) */
```

Definition at line 17 of file [embroidery.h](#).

4.5.1.105 LIBEMBROIDERY_EMBEDDED_VERSION

```
#define LIBEMBROIDERY_EMBEDDED_VERSION 0
```

Definition at line 9 of file [embroidery.h](#).

4.5.1.106 Madeira_Polyester

```
#define Madeira_Polyester 11
```

Definition at line 98 of file [embroidery.h](#).

4.5.1.107 Madeira_Rayon

```
#define Madeira_Rayon 12
```

Definition at line 99 of file [embroidery.h](#).

4.5.1.108 Marathon_Polyester

```
#define Marathon_Polyester 9
```

Definition at line 96 of file [embroidery.h](#).

4.5.1.109 Marathon_Rayon

```
#define Marathon_Rayon 10
```

Definition at line 97 of file [embroidery.h](#).

4.5.1.110 MAX_STITCHES

```
#define MAX_STITCHES 1000000
```

Definition at line 154 of file [embroidery.h](#).

4.5.1.111 MAX_THREADS

```
#define MAX_THREADS 256
```

Definition at line 149 of file [embroidery.h](#).

4.5.1.112 Metro_Polyester

```
#define Metro_Polyester 13
```

Definition at line 100 of file [embroidery.h](#).

4.5.1.113 NORMAL

```
#define NORMAL 0 /* stitch to (x, y) */
```

Definition at line 16 of file [embroidery.h](#).

4.5.1.114 numberOfFormats

```
#define numberOfFormats 61
```

Definition at line 144 of file [embroidery.h](#).

4.5.1.115 Pantone

```
#define Pantone 14
```

Definition at line 101 of file [embroidery.h](#).

4.5.1.116 **pcm_thread**

```
#define pcm_thread 26
```

Definition at line 113 of file [embroidery.h](#).

4.5.1.117 **pec_thread**

```
#define pec_thread 27
```

Definition at line 114 of file [embroidery.h](#).

4.5.1.118 **RobisonAnton_Polyester**

```
#define RobisonAnton_Polyester 15
```

Definition at line 102 of file [embroidery.h](#).

4.5.1.119 **RobisonAnton_Rayon**

```
#define RobisonAnton_Rayon 16
```

Definition at line 103 of file [embroidery.h](#).

4.5.1.120 **SEQUIN**

```
#define SEQUIN 8 /* sequin */
```

Definition at line 20 of file [embroidery.h](#).

4.5.1.121 **shv_thread**

```
#define shv_thread 28
```

Definition at line 115 of file [embroidery.h](#).

4.5.1.122 Sigma_Polyester

```
#define Sigma_Polyester 17
```

Definition at line 104 of file [embroidery.h](#).

4.5.1.123 STOP

```
#define STOP 4 /* pause machine for thread change */
```

Definition at line 19 of file [embroidery.h](#).

4.5.1.124 Sulky_Rayon

```
#define Sulky_Rayon 18
```

Definition at line 105 of file [embroidery.h](#).

4.5.1.125 SVG_Colors

```
#define SVG_Colors 23
```

Definition at line 110 of file [embroidery.h](#).

4.5.1.126 ThreadArt_Polyester

```
#define ThreadArt_Polyester 20
```

Definition at line 107 of file [embroidery.h](#).

4.5.1.127 ThreadArt_Rayon

```
#define ThreadArt_Rayon 19
```

Definition at line 106 of file [embroidery.h](#).

4.5.1.128 ThreaDelight_Polyester

```
#define ThreaDelight_Polyester 21
```

Definition at line 108 of file [embroidery.h](#).

4.5.1.129 TRIM

```
#define TRIM 2 /* trim + move to (x, y) */
```

Definition at line 18 of file [embroidery.h](#).

4.5.1.130 Z102_Isacord_Polyester

```
#define Z102_Isacord_Polyester 22
```

Definition at line 109 of file [embroidery.h](#).

4.5.2 Typedef Documentation

4.5.2.1 EmbAlignedDim

```
typedef struct EmbAlignedDim_ EmbAlignedDim
```

4.5.2.2 EmbAngularDim

```
typedef struct EmbAngularDim_ EmbAngularDim
```

4.5.2.3 EmbArc

```
typedef struct EmbArc_ EmbArc
```

4.5.2.4 EmbArcLengthDim

```
typedef struct EmbArcLengthDim_ EmbArcLengthDim
```

4.5.2.5 EmbArray

```
typedef struct EmbArray_ EmbArray
```

The basic array type.

Definition at line 203 of file [embroidery.h](#).

4.5.2.6 EmbBezier

```
typedef struct EmbBezier_ EmbBezier
```

4.5.2.7 EmbBlock

```
typedef struct EmbBlock_ EmbBlock
```

4.5.2.8 EmbCircle

```
typedef struct EmbCircle_ EmbCircle
```

4.5.2.9 EmbColor

```
typedef struct EmbColor_ EmbColor
```

EmbColor uses the light primaries: red, green, blue in that order.

4.5.2.10 EmbDiameterDim

```
typedef struct EmbDiameterDim_ EmbDiameterDim
```

4.5.2.11 EmbEllipse

```
typedef struct EmbEllipse_ EmbEllipse
```

4.5.2.12 EmbFlag

```
typedef int EmbFlag
```

Definition at line 351 of file [embroidery.h](#).

4.5.2.13 EmbFormatList

```
typedef struct EmbFormatList_ EmbFormatList
```

4.5.2.14 EmbGeometry

```
typedef struct EmbGeometry_ EmbGeometry
```

4.5.2.15 EmbImage

```
typedef struct EmbImage_ EmbImage
```

4.5.2.16 EmbInfiniteLine

```
typedef struct EmbInfiniteLine_ EmbInfiniteLine
```

4.5.2.17 EmbLayer

```
typedef struct EmbLayer_ EmbLayer
```

4.5.2.18 EmbLeaderDim

```
typedef struct EmbLeaderDim_ EmbLeaderDim
```

4.5.2.19 EmbLine

```
typedef struct EmbLine_ EmbLine
```

4.5.2.20 EmbLinearDim

```
typedef struct EmbLinearDim_ EmbLinearDim
```

4.5.2.21 EmbOrdinateDim

```
typedef struct EmbOrdinateDim_ EmbOrdinateDim
```

4.5.2.22 EmbPath

```
typedef struct EmbPath_ EmbPath
```

4.5.2.23 EmbPattern

```
typedef struct EmbPattern_ EmbPattern
```

4.5.2.24 EmbPoint

```
typedef struct EmbPoint_ EmbPoint
```

4.5.2.25 EmbPolygon

```
typedef EmbPath EmbPolygon
```

Definition at line 349 of file [embroidery.h](#).

4.5.2.26 EmbPolyline

```
typedef EmbPath EmbPolyline
```

Definition at line 350 of file [embroidery.h](#).

4.5.2.27 EmbRadiusDim

```
typedef struct EmbRadiusDim_ EmbRadiusDim
```

4.5.2.28 EmbRay

```
typedef struct EmbRay_ EmbRay
```

4.5.2.29 EmbReal

```
typedef float EmbReal
```

Definition at line 177 of file [embroidery.h](#).

4.5.2.30 EmbRect

```
typedef struct EmbRect_ EmbRect
```

4.5.2.31 EmbSatinOutline

```
typedef struct EmbSatinOutline_ EmbSatinOutline
```

4.5.2.32 EmbSpline

```
typedef struct EmbSpline_ EmbSpline
```

4.5.2.33 EmbStitch

```
typedef struct EmbStitch_ EmbStitch
```

4.5.2.34 EmbTextMulti

```
typedef struct EmbTextMulti_ EmbTextMulti
```

4.5.2.35 EmbTextSingle

```
typedef struct EmbTextSingle_ EmbTextSingle
```

4.5.2.36 EmbThread

```
typedef struct EmbThread_ EmbThread
```

4.5.2.37 EmbTime

```
typedef struct EmbTime_ EmbTime
```

4.5.2.38 EmbVector

```
typedef struct EmbVector_ EmbVector
```

The basic type to represent points absolutely or represent directions.

Positive y is up, units are in mm.

4.5.2.39 L_system

```
typedef struct LSYSTEM L_system
```

4.5.2.40 thread_color

```
typedef struct thread_color_ thread_color
```

4.5.3 Function Documentation

4.5.3.1 convert()

```
EMB_PUBLIC int convert (
    const char * inf,
    const char * outf )
```

Definition at line 1108 of file [pattern.c](#).

4.5.3.2 degrees()

```
EMB_PUBLIC EmbReal degrees (
    EmbReal radian )
```

4.5.3.3 emb_identify_format()

```
EMB_PUBLIC int emb_identify_format (
    const char * ending )
```

Definition at line 167 of file [formats.c](#).

4.5.3.4 emb_round()

```
EMB_PUBLIC int emb_round (
    EmbReal x )
```

4.5.3.5 embArc_clockwise()

```
EMB_PUBLIC char embArc_clockwise (
    EmbArc arc )
```

4.5.3.6 embArc_init()

```
EMB_PUBLIC EmbArc embArc_init (
    void )
```

4.5.3.7 embArray_addArc()

```
EMB_PUBLIC int embArray_addArc (
    EmbArray * g,
    EmbArc arc )
```

Definition at line 96 of file [array.c](#).

4.5.3.8 embArray_addCircle()

```
EMB_PUBLIC int embArray_addCircle (
    EmbArray * g,
    EmbCircle circle )
```

Definition at line 108 of file [array.c](#).

4.5.3.9 embArray_addEllipse()

```
EMB_PUBLIC int embArray_addEllipse (
    EmbArray * g,
    EmbEllipse ellipse )
```

Definition at line 120 of file [array.c](#).

4.5.3.10 embArray_addFlag()

```
EMB_PUBLIC int embArray_addFlag (
    EmbArray * g,
    int flag )
```

Definition at line 132 of file [array.c](#).

4.5.3.11 embArray_addLine()

```
EMB_PUBLIC int embArray_addLine (
    EmbArray * g,
    EmbLine line )
```

Definition at line 144 of file [array.c](#).

4.5.3.12 embArray_addPath()

```
EMB_PUBLIC int embArray_addPath (
    EmbArray * g,
    EmbPath p )
```

Definition at line 156 of file [array.c](#).

4.5.3.13 embArray_addPoint()

```
EMB_PUBLIC int embArray_addPoint (
    EmbArray * g,
    EmbPoint p )
```

Definition at line 168 of file [array.c](#).

4.5.3.14 embArray_addPolygon()

```
EMB_PUBLIC int embArray_addPolygon (
    EmbArray * g,
    EmbPolygon p )
```

Definition at line 192 of file [array.c](#).

4.5.3.15 embArray_addPolyline()

```
EMB_PUBLIC int embArray_addPolyline (
    EmbArray * g,
    EmbPolyline p )
```

Definition at line 180 of file [array.c](#).

4.5.3.16 embArray_addRect()

```
EMB_PUBLIC int embArray_addRect (
    EmbArray * g,
    EmbRect rect )
```

Definition at line 204 of file [array.c](#).

4.5.3.17 embArray_addStitch()

```
EMB_PUBLIC int embArray_addStitch (
    EmbArray * g,
    EmbStitch st )
```

Definition at line 216 of file [array.c](#).

4.5.3.18 embArray_addThread()

```
EMB_PUBLIC int embArray_addThread (
    EmbArray * g,
    EmbThread p )
```

4.5.3.19 embArray_addVector()

```
EMB_PUBLIC int embArray_addVector (
    EmbArray * g,
    EmbVector b )
```

Definition at line 227 of file [array.c](#).

4.5.3.20 embArray_copy()

```
EMB_PUBLIC void embArray_copy (
    EmbArray * dst,
    EmbArray * src )
```

Definition at line 74 of file [array.c](#).

4.5.3.21 embArray_create()

```
EMB_PUBLIC EmbArray * embArray_create (
    int type )
```

Definition at line 19 of file [array.c](#).

4.5.3.22 embArray_free()

```
EMB_PUBLIC void embArray_free (
    EmbArray * p )
```

Definition at line 239 of file [array.c](#).

4.5.3.23 embArray_resize()

```
EMB_PUBLIC int embArray_resize (
    EmbArray * g )
```

Definition at line 41 of file [array.c](#).

4.5.3.24 embCircle_init()

```
EMB_PUBLIC EmbCircle embCircle_init (
    void )
```

4.5.3.25 embColor_create()

```
EMB_PUBLIC EmbColor * embColor_create (
    unsigned char r,
    unsigned char g,
    unsigned char b )
```

4.5.3.26 embColor_distance()

```
EMB_PUBLIC int embColor_distance (
    EmbColor a,
    EmbColor b )
```

Definition at line 671 of file [main.c](#).

4.5.3.27 embColor_fromHexStr()

```
EMB_PUBLIC EmbColor embColor_fromHexStr (
    char * val )
```

Definition at line 32 of file [encoding.c](#).

4.5.3.28 embColor_make()

```
EMB_PUBLIC EmbColor embColor_make (
    unsigned char r,
    unsigned char g,
    unsigned char b )
```

4.5.3.29 embEllipse_area()

```
EMB_PUBLIC EmbReal embEllipse_area (
    EmbEllipse ellipse )
```

4.5.3.30 embEllipse_diameterX()

```
EMB_PUBLIC EmbReal embEllipse_diameterX (
    EmbEllipse ellipse )
```

4.5.3.31 embEllipse_diameterY()

```
EMB_PUBLIC EmbReal embEllipse_diameterY (
    EmbEllipse ellipse )
```

4.5.3.32 embEllipse_height()

```
EMB_PUBLIC EmbReal embEllipse_height (
    EmbEllipse ellipse )
```

4.5.3.33 embEllipse_init()

```
EMB_PUBLIC EmbEllipse embEllipse_init (
    void )
```

4.5.3.34 embEllipse_make()

```
EMB_PUBLIC EmbEllipse embEllipse_make (
    EmbReal cx,
    EmbReal cy,
    EmbReal rx,
    EmbReal ry )
```

4.5.3.35 embEllipse_perimeter()

```
EMB_PUBLIC EmbReal embEllipse_perimeter (
    EmbEllipse ellipse )
```

4.5.3.36 embEllipse_width()

```
EMB_PUBLIC EmbReal embEllipse_width (
    EmbEllipse ellipse )
```

4.5.3.37 embGeometry_boundingRect()

```
EMB_PUBLIC EmbRect embGeometry_boundingRect (
    EmbGeometry * obj )
```

Definition at line 102 of file [geometry.c](#).

4.5.3.38 embGeometry_free()

```
EMB_PUBLIC void embGeometry_free (
    EmbGeometry * obj )
```

Definition at line 63 of file [geometry.c](#).

4.5.3.39 embGeometry_init()

```
EMB_PUBLIC EmbGeometry * embGeometry_init (
    int type_in )
```

Definition at line 20 of file [geometry.c](#).

4.5.3.40 embGeometry_move()

```
EMB_PUBLIC void embGeometry_move (
    EmbGeometry * obj,
    EmbVector delta )
```

Definition at line 81 of file [geometry.c](#).

4.5.3.41 embGeometry_vulcanize()

```
EMB_PUBLIC void embGeometry_vulcanize (
    EmbGeometry * obj )
```

Definition at line 128 of file [geometry.c](#).

4.5.3.42 embImage_create()

```
EMB_PUBLIC EmbImage embImage_create (
    int ,
    int )
```

4.5.3.43 embImage_free()

```
EMB_PUBLIC void embImage_free (
    EmbImage * image )
```

4.5.3.44 embImage_read()

```
EMB_PUBLIC void embImage_read (
    EmbImage * image,
    char * fname )
```

4.5.3.45 embImage_write()

```
EMB_PUBLIC int embImage_write (
    EmbImage * image,
    char * fname )
```

4.5.3.46 embLine_intersectionPoint()

```
EMB_PUBLIC EmbVector embLine_intersectionPoint (
    EmbLine line1,
    EmbLine line2 )
```

4.5.3.47 embLine_make()

```
EMB_PUBLIC EmbLine embLine_make (
    EmbReal x1,
    EmbReal y1,
    EmbReal x2,
    EmbReal y2 )
```

4.5.3.48 embLine_normalVector()

```
EMB_PUBLIC void embLine_normalVector (
    EmbLine line,
    EmbVector * result,
    int clockwise )
```

4.5.3.49 embPattern_addCircleAbs()

```
EMB_PUBLIC void embPattern_addCircleAbs (
    EmbPattern * p,
    EmbCircle circle )
```

Adds a circle object to pattern (*p*) with its center at the absolute position (*cx,cy*) with a radius of (*r*). Positive y is up.
Units are in millimeters.

Definition at line 787 of file [pattern.c](#).

4.5.3.50 embPattern_addEllipseAbs()

```
EMB_PUBLIC void embPattern_addEllipseAbs (
    EmbPattern * p,
    EmbEllipse ellipse )
```

Adds an ellipse object to pattern (*p*) with its center at the absolute position (*cx,cy*) with radii of (*rx,ry*). Positive y is up. Units are in millimeters.

Definition at line 801 of file [pattern.c](#).

4.5.3.51 embPattern_addLineAbs()

```
EMB_PUBLIC void embPattern_addLineAbs (
    EmbPattern * p,
    EmbLine line )
```

Adds a line object to pattern (*p*) starting at the absolute position (*x1,y1*) and ending at the absolute position (*x2,y2*). Positive y is up. Units are in millimeters.

Definition at line 816 of file [pattern.c](#).

4.5.3.52 embPattern_addPathAbs()

```
EMB_PUBLIC void embPattern_addPathAbs (
    EmbPattern * p,
    EmbPath obj )
```

Definition at line 827 of file [pattern.c](#).

4.5.3.53 embPattern_addPointAbs()

```
EMB_PUBLIC void embPattern_addPointAbs (
    EmbPattern * p,
    EmbPoint obj )
```

Adds a point object to pattern (*p*) at the absolute position (*x,y*). Positive y is up. Units are in millimeters.

Definition at line 843 of file [pattern.c](#).

4.5.3.54 embPattern_addPolygonAbs()

```
EMB_PUBLIC void embPattern_addPolygonAbs (
    EmbPattern * p,
    EmbPolygon obj )
```

Definition at line [854](#) of file [pattern.c](#).

4.5.3.55 embPattern_addPolylineAbs()

```
EMB_PUBLIC void embPattern_addPolylineAbs (
    EmbPattern * p,
    EmbPolyline obj )
```

4.5.3.56 embPattern_addRectAbs()

```
EMB_PUBLIC void embPattern_addRectAbs (
    EmbPattern * p,
    EmbRect rect )
```

Adds a rectangle object to pattern (*p*) at the absolute position (*x,y*) with a width of (*w*) and a height of (*h*). Positive y is up. Units are in millimeters.

Definition at line [888](#) of file [pattern.c](#).

4.5.3.57 embPattern_addStitchAbs()

```
EMB_PUBLIC void embPattern_addStitchAbs (
    EmbPattern * p,
    EmbReal x,
    EmbReal y,
    int flags,
    int isAutoColorIndex )
```

Adds a stitch to the pattern (*p*) at the absolute position (*x,y*). Positive y is up. Units are in millimeters.

Definition at line [236](#) of file [pattern.c](#).

4.5.3.58 embPattern_addStitchRel()

```
EMB_PUBLIC void embPattern_addStitchRel (
    EmbPattern * p,
    EmbReal dx,
    EmbReal dy,
    int flags,
    int isAutoColorIndex )
```

Adds a stitch to the pattern (*p*) at the relative position (*dx,dy*) to the previous stitch. Positive y is up. Units are in millimeters.

Definition at line 290 of file [pattern.c](#).

4.5.3.59 embPattern_addThread()

```
EMB_PUBLIC int embPattern_addThread (
    EmbPattern * p,
    EmbThread thread )
```

Definition at line 66 of file [pattern.c](#).

4.5.3.60 embPattern_calcBoundingBox()

```
EMB_PUBLIC EmbRect embPattern_calcBoundingBox (
    EmbPattern * p )
```

Returns an EmbRect that encapsulates all stitches and objects in the pattern (*p*).

Definition at line 340 of file [pattern.c](#).

4.5.3.61 embPattern_center()

```
EMB_PUBLIC void embPattern_center (
    EmbPattern * p )
```

Definition at line 709 of file [pattern.c](#).

4.5.3.62 embPattern_changeColor()

```
EMB_PUBLIC void embPattern_changeColor (
    EmbPattern * p,
    int index )
```

Definition at line 312 of file [pattern.c](#).

4.5.3.63 embPattern_color_count()

```
EMB_PUBLIC int embPattern_color_count (
    EmbPattern * pattern,
    EmbColor startColor )
```

Definition at line 911 of file [pattern.c](#).

4.5.3.64 embPattern_combine()

```
EMB_PUBLIC EmbPattern * embPattern_combine (
    EmbPattern * p1,
    EmbPattern * p2 )
```

Definition at line 854 of file [fill.c](#).

4.5.3.65 embPattern_combineJumpStitches()

```
EMB_PUBLIC void embPattern_combineJumpStitches (
    EmbPattern * p )
```

Definition at line 624 of file [pattern.c](#).

4.5.3.66 embPattern_convertGeometry()

```
EMB_PUBLIC void embPattern_convertGeometry (
    EmbPattern * p )
```

Definition at line 987 of file [fill.c](#).

4.5.3.67 embPattern_copyPolylinesToStitchList()

```
EMB_PUBLIC void embPattern_copyPolylinesToStitchList (
    EmbPattern * pattern )
```

4.5.3.68 embPattern_copyStitchListToPolylines()

```
EMB_PUBLIC void embPattern_copyStitchListToPolylines (
    EmbPattern * pattern )
```

4.5.3.69 embPattern_correctForMaxStitchLength()

```
EMB_PUBLIC void embPattern_correctForMaxStitchLength (
    EmbPattern * p,
    EmbReal maxStitchLength,
    EmbReal maxJumpLength )
```

Definition at line 660 of file [pattern.c](#).

4.5.3.70 embPattern_create()

```
EMB_PUBLIC EmbPattern * embPattern_create (
    void )
```

Returns a pointer to an EmbPattern. It is created on the heap. The caller is responsible for freeing the allocated memory with [embPattern_free\(\)](#).

Definition at line 21 of file [pattern.c](#).

4.5.3.71 embPattern_crossstitch()

```
EMB_PUBLIC void embPattern_crossstitch (
    EmbPattern * pattern,
    EmbImage * image,
    int threshold )
```

Definition at line 239 of file [fill.c](#).

4.5.3.72 embPattern_designDetails()

```
EMB_PUBLIC void embPattern_designDetails (
    EmbPattern * p )
```

Definition at line 942 of file [pattern.c](#).

4.5.3.73 embPattern_end()

```
EMB_PUBLIC void embPattern_end (
    EmbPattern * p )
```

Definition at line 898 of file [pattern.c](#).

4.5.3.74 embPattern_fixColorCount()

```
EMB_PUBLIC void embPattern_fixColorCount (
    EmbPattern * p )
```

Definition at line [79](#) of file [pattern.c](#).

4.5.3.75 embPattern_flip()

```
EMB_PUBLIC void embPattern_flip (
    EmbPattern * p,
    int horz,
    int vert )
```

Flips the entire pattern (*p*) horizontally about the x-axis if (*horz*) is true. Flips the entire pattern (*p*) vertically about the y-axis if (*vert*) is true.

Definition at line [497](#) of file [pattern.c](#).

4.5.3.76 embPattern_flipHorizontal()

```
EMB_PUBLIC void embPattern_flipHorizontal (
    EmbPattern * p )
```

Flips the entire pattern (*p*) horizontally about the y-axis.

Definition at line [472](#) of file [pattern.c](#).

4.5.3.77 embPattern_flipVertical()

```
EMB_PUBLIC void embPattern_flipVertical (
    EmbPattern * p )
```

Flips the entire pattern (*p*) vertically about the x-axis.

Definition at line [484](#) of file [pattern.c](#).

4.5.3.78 embPattern_free()

```
EMB_PUBLIC void embPattern_free (
    EmbPattern * p )
```

Frees all memory allocated in the pattern (*p*).

Definition at line [771](#) of file [pattern.c](#).

4.5.3.79 embPattern_hideStitchesOverLength()

```
EMB_PUBLIC void embPattern_hideStitchesOverLength (
    EmbPattern * p,
    int length )
```

Definition at line [42](#) of file [pattern.c](#).

4.5.3.80 embPattern_horizontal_fill()

```
EMB_PUBLIC void embPattern_horizontal_fill (
    EmbPattern * pattern,
    EmbImage * image,
    int threshhold )
```

Definition at line [212](#) of file [fill.c](#).

4.5.3.81 embPattern_jumpStitches()

```
EMB_PUBLIC int embPattern_jumpStitches (
    EmbPattern * pattern )
```

Definition at line [1240](#) of file [pattern.c](#).

4.5.3.82 embPattern_lengthHistogram()

```
EMB_PUBLIC void embPattern_lengthHistogram (
    EmbPattern * pattern,
    int * bin,
    int NUMBINS )
```

Definition at line [1205](#) of file [pattern.c](#).

4.5.3.83 embPattern_loadExternalColorFile()

```
EMB_PUBLIC void embPattern_loadExternalColorFile (
    EmbPattern * p,
    const char * fileName )
```

Definition at line [732](#) of file [pattern.c](#).

4.5.3.84 embPattern_maximumStitchLength()

```
EMB_PUBLIC EmbReal embPattern_maximumStitchLength (
    EmbPattern * pattern )
```

Definition at line 1184 of file [pattern.c](#).

4.5.3.85 embPattern_minimumStitchLength()

```
EMB_PUBLIC EmbReal embPattern_minimumStitchLength (
    EmbPattern * pattern )
```

Definition at line 1163 of file [pattern.c](#).

4.5.3.86 embPattern_movePolylinesToStitchList()

```
EMB_PUBLIC void embPattern_movePolylinesToStitchList (
    EmbPattern * pattern )
```

4.5.3.87 embPattern_moveStitchListToPolylines()

```
EMB_PUBLIC void embPattern_moveStitchListToPolylines (
    EmbPattern * pattern )
```

4.5.3.88 embPattern_read()

```
EMB_PUBLIC char embPattern_read (
    EmbPattern * pattern,
    const char * fileName,
    int format )
```

Definition at line 259 of file [formats.c](#).

4.5.3.89 embPattern_readAuto()

```
EMB_PUBLIC char embPattern_readAuto (
    EmbPattern * pattern,
    const char * fileName )
```

Definition at line 706 of file [formats.c](#).

4.5.3.90 embPattern_realStitches()

```
EMB_PUBLIC int embPattern_realStitches (
    EmbPattern * pattern )
```

Definition at line 1227 of file [pattern.c](#).

4.5.3.91 embPattern_render()

```
EMB_PUBLIC int embPattern_render (
    EmbPattern * pattern,
    char * fname )
```

4.5.3.92 embPattern_scale()

```
EMB_PUBLIC void embPattern_scale (
    EmbPattern * p,
    EmbReal scale )
```

Definition at line 324 of file [pattern.c](#).

4.5.3.93 embPattern_simulate()

```
EMB_PUBLIC int embPattern_simulate (
    EmbPattern * pattern,
    char * fname )
```

4.5.3.94 embPattern_totalStitchLength()

```
EMB_PUBLIC EmbReal embPattern_totalStitchLength (
    EmbPattern * pattern )
```

Definition at line 1143 of file [pattern.c](#).

4.5.3.95 embPattern_trimStitches()

```
EMB_PUBLIC int embPattern_trimStitches (
    EmbPattern * pattern )
```

Definition at line 1254 of file [pattern.c](#).

4.5.3.96 embPattern_write()

```
EMB_PUBLIC char embPattern_write (
    EmbPattern * pattern,
    const char * fileName,
    int format )
```

Definition at line [480](#) of file [formats.c](#).

4.5.3.97 embPattern_writeAuto()

```
EMB_PUBLIC char embPattern_writeAuto (
    EmbPattern * pattern,
    const char * fileName )
```

Definition at line [718](#) of file [formats.c](#).

4.5.3.98 embRect_area()

```
EMB_PUBLIC EmbReal embRect_area (
    EmbRect )
```

4.5.3.99 embRect_init()

```
EMB_PUBLIC EmbRect embRect_init (
    void )
```

4.5.3.100 embSatinOutline_generateSatinOutline()

```
EMB_PUBLIC void embSatinOutline_generateSatinOutline (
    EmbArray * lines,
    EmbReal thickness,
    EmbSatinOutline * result )
```

Definition at line [520](#) of file [main.c](#).

4.5.3.101 embSatinOutline_renderStitches()

```
EMB_PUBLIC EmbArray * embSatinOutline_renderStitches (
    EmbSatinOutline * result,
    EmbReal density )
```

Definition at line 608 of file [main.c](#).

4.5.3.102 embThread_findNearestColor()

```
EMB_PUBLIC int embThread_findNearestColor (
    EmbColor color,
    EmbColor * color_list,
    int n_colors )
```

Returns the closest color to the required color based on a list of available threads. The algorithm is a simple least squares search against the list. If the (square of) Euclidean 3-dimensional distance between the points in (red, green, blue) space is smaller then the index is saved and the remaining index is returned to the caller.

Parameters

<i>color</i>	The EmbColor color to match.
<i>colors</i>	The EmbThreadList pointer to start the search at.
<i>mode</i>	Is the argument an array of threads (0) or colors (1)?

Returns

closestIndex The entry in the ThreadList that matches.

Definition at line 715 of file [main.c](#).

4.5.3.103 embThread_findNearestThread()

```
EMB_PUBLIC int embThread_findNearestThread (
    EmbColor color,
    EmbThread * threads,
    int n_threads )
```

Definition at line 731 of file [main.c](#).

4.5.3.104 embThread_getRandom()

```
EMB_PUBLIC EmbThread embThread_getRandom (
    void )
```

Returns a random thread color, useful in filling in cases where the actual color of the thread doesn't matter but one needs to be declared to test or render a pattern.

Returns

c The resulting color.

Definition at line [754](#) of file [main.c](#).

4.5.3.105 embTime_initNow()

```
EMB_PUBLIC void embTime_initNow (
    EmbTime * t )
```

Definition at line [885](#) of file [main.c](#).

4.5.3.106 embTime_time()

```
EMB_PUBLIC EmbTime embTime_time (
    EmbTime * t )
```

Definition at line [905](#) of file [main.c](#).

4.5.3.107 embVector_add()

```
EMB_PUBLIC EmbVector embVector_add (
    EmbVector v1,
    EmbVector v2 )
```

4.5.3.108 embVector_angle()

```
EMB_PUBLIC EmbReal embVector_angle (
    EmbVector v )
```

4.5.3.109 embVector_average()

```
EMB_PUBLIC EmbVector embVector_average (
    EmbVector v1,
    EmbVector v2 )
```

4.5.3.110 embVector_cross()

```
EMB_PUBLIC EmbReal embVector_cross (
    EmbVector v1,
    EmbVector v2 )
```

4.5.3.111 embVector_distance()

```
EMB_PUBLIC EmbReal embVector_distance (
    EmbVector a,
    EmbVector b )
```

4.5.3.112 embVector_dot()

```
EMB_PUBLIC EmbReal embVector_dot (
    EmbVector v1,
    EmbVector v2 )
```

4.5.3.113 embVector_length()

```
EMB_PUBLIC EmbReal embVector_length (
    EmbVector vector )
```

4.5.3.114 embVector_multiply()

```
EMB_PUBLIC void embVector_multiply (
    EmbVector vector,
    EmbReal magnitude,
    EmbVector * result )
```

4.5.3.115 embVector_normalize()

```
EMB_PUBLIC void embVector_normalize (
    EmbVector vector,
    EmbVector * result )
```

4.5.3.116 embVector_relativeX()

```
EMB_PUBLIC EmbReal embVector_relativeX (
    EmbVector a1,
    EmbVector a2,
    EmbVector a3 )
```

4.5.3.117 embVector_relativeY()

```
EMB_PUBLIC EmbReal embVector_relativeY (
    EmbVector a1,
    EmbVector a2,
    EmbVector a3 )
```

4.5.3.118 embVector_subtract()

```
EMB_PUBLIC EmbVector embVector_subtract (
    EmbVector v1,
    EmbVector v2 )
```

4.5.3.119 embVector_transpose_product()

```
EMB_PUBLIC void embVector_transpose_product (
    EmbVector v1,
    EmbVector v2,
    EmbVector * result )
```

4.5.3.120 embVector_unit()

```
EMB_PUBLIC EmbVector embVector_unit (
    EmbReal angle )
```

4.5.3.121 full_test_matrix()

```
EMB_PUBLIC int full_test_matrix (
    char * fname )
```

4.5.3.122 getArcCenter()

```
EMB_PUBLIC void getArcCenter (
    EmbArc arc,
    EmbVector * arcCenter )
```

4.5.3.123 getArcDataFromBulge()

```
EMB_PUBLIC char getArcDataFromBulge (
    EmbReal bulge,
    EmbArc * arc,
    EmbReal * arcCenterX,
    EmbReal * arcCenterY,
    EmbReal * radius,
    EmbReal * diameter,
    EmbReal * chord,
    EmbReal * chordMidX,
    EmbReal * chordMidY,
    EmbReal * sagitta,
    EmbReal * apothem,
    EmbReal * incAngleInDegrees,
    char * clockwise )
```

4.5.3.124 getCircleCircleIntersections()

```
EMB_PUBLIC int getCircleCircleIntersections (
    EmbCircle c0,
    EmbCircle c1,
    EmbVector * v0,
    EmbVector * v1 )
```

4.5.3.125 getCircleTangentPoints()

```
EMB_PUBLIC int getCircleTangentPoints (
    EmbCircle c,
    EmbVector p,
    EmbVector * v0,
    EmbVector * v1 )
```

4.5.3.126 hilbert_curve()

```
EMB_PUBLIC int hilbert_curve (
    EmbPattern * pattern,
    int iterations )
```

Definition at line [268](#) of file [fill.c](#).

4.5.3.127 lindenmayer_system()

```
EMB_PUBLIC int lindenmayer_system (
    L_system L,
    char * state,
    int iteration,
    int complete )
```

Definition at line [27](#) of file [fill.c](#).

4.5.3.128 radians()

```
EMB_PUBLIC EmbReal radians (
    EmbReal degree )
```

4.5.3.129 report()

```
EMB_PUBLIC void report (
    int result,
    char * label )
```

4.5.3.130 testMain()

```
EMB_PUBLIC void testMain (
    int level )
```

4.5.3.131 threadColor()

```
EMB_PUBLIC int threadColor (
    const char * name,
    int brand )
```

Definition at line [4012](#) of file [thread-color.c](#).

4.5.3.132 threadColorName()

```
EMB_PUBLIC const char * threadColorName (
    unsigned int color,
    int brand )
```

Definition at line 4035 of file [thread-color.c](#).

4.5.3.133 threadColorNum()

```
EMB_PUBLIC int threadColorNum (
    unsigned int color,
    int brand )
```

Definition at line 4023 of file [thread-color.c](#).

4.5.4 Variable Documentation

4.5.4.1 _dxfColorTable

```
const unsigned char _dxfColorTable[ ][3] [extern]
```

Definition at line 14 of file [thread-color.c](#).

4.5.4.2 black_thread

```
EmbThread black_thread [extern]
```

Definition at line 56 of file [main.c](#).

4.5.4.3 emb_error

```
int emb_error [extern]
```

Definition at line 58 of file [main.c](#).

4.5.4.4 emb_verbose

```
int emb_verbose [extern]
```

Definition at line 57 of file [main.c](#).

4.5.4.5 embConstantPi

```
const EmbReal embConstantPi [extern]
```

Definition at line 60 of file [main.c](#).

4.5.4.6 formatTable

```
EmbFormatList formatTable[numberOfFormats] [extern]
```

Definition at line 21 of file [formats.c](#).

4.5.4.7 husThreads

```
const EmbThread husThreads[] [extern]
```

Definition at line 15 of file [thread-color.c](#).

4.5.4.8 jefThreads

```
const EmbThread jefThreads[] [extern]
```

Definition at line 16 of file [thread-color.c](#).

4.5.4.9 pcmThreads

```
const EmbThread pcmThreads[] [extern]
```

Definition at line 18 of file [thread-color.c](#).

4.5.4.10 pecThreadCount

```
const int pecThreadCount [extern]
```

Definition at line 21 of file [thread-color.c](#).

4.5.4.11 pecThreads

```
const EmbThread pecThreads[] [extern]
```

Definition at line 19 of file [thread-color.c](#).

4.5.4.12 shvThreadCount

```
const int shvThreadCount [extern]
```

Definition at line 20 of file [thread-color.c](#).

4.5.4.13 shvThreads

```
const EmbThread shvThreads[] [extern]
```

Definition at line 17 of file [thread-color.c](#).

4.5.4.14 vipDecodingTable

```
const unsigned char vipDecodingTable[] [extern]
```

4.6 embroidery.h

[Go to the documentation of this file.](#)

```

00001 #ifndef LIBEMBROIDERY_HEADER_
00002 #define LIBEMBROIDERY_HEADER_
00003
00004 #ifdef __cplusplus
00005 extern "C" {
00006 #endif
00007
00008 #ifndef LIBEMBROIDERY_EMBEDDED_VERSION
00009 #define LIBEMBROIDERY_EMBEDDED_VERSION 0
00010 #endif
00011
00012 /* MACROS
00013 *****/
00014
00015 /* Machine codes for stitch flags */
00016 #define NORMAL 0 /* stitch to (x, y) */
00017 #define JUMP 1 /* move to (x, y) */
00018 #define TRIM 2 /* trim + move to (x, y) */
00019 #define STOP 4 /* pause machine for thread change */
00020 #define SEQUIN 8 /* sequin */
00021 #define END 16 /* end of program */
00022
00023 /* Format identifiers */
00024 #define EMB_FORMAT_100 0
00025 #define EMB_FORMAT_100 1
00026 #define EMB_FORMAT_ART 2
00027 #define EMB_FORMAT_BMC 3
00028 #define EMB_FORMAT_BRO 4
00029 #define EMB_FORMAT_CND 5
00030 #define EMB_FORMAT_COL 6
00031 #define EMB_FORMAT_CSD 7
00032 #define EMB_FORMAT_CSV 8
00033 #define EMB_FORMAT_DAT 9
00034 #define EMB_FORMAT_DEM 10
00035 #define EMB_FORMAT_DSB 11
00036 #define EMB_FORMAT_DST 12
00037 #define EMB_FORMAT_DSZ 13
00038 #define EMB_FORMAT_DXF 14
00039 #define EMB_FORMAT_EDR 15
00040 #define EMB_FORMAT_EMD 16
00041 #define EMB_FORMAT_EXP 17
00042 #define EMB_FORMAT_EXY 18
00043 #define EMB_FORMAT_EYS 19
00044 #define EMB_FORMAT_FXY 20
00045 #define EMB_FORMAT_GC 21
00046 #define EMB_FORMAT_GNC 22
00047 #define EMB_FORMAT_GT 23
00048 #define EMB_FORMAT_HUS 24
00049 #define EMB_FORMAT_INB 25
00050 #define EMB_FORMAT_INF 26
00051 #define EMB_FORMAT_JEF 27
00052 #define EMB_FORMAT_KSM 28
00053 #define EMB_FORMAT_MAX 29
00054 #define EMB_FORMAT_MIT 30
00055 #define EMB_FORMAT_NEW 31
00056 #define EMB_FORMAT_OFM 32
00057 #define EMB_FORMAT_PCD 33
00058 #define EMB_FORMAT_PCM 34
00059 #define EMB_FORMAT_PCQ 35
00060 #define EMB_FORMAT_PCS 36
00061 #define EMB_FORMAT_PEC 37
00062 #define EMB_FORMAT_PEL 38
00063 #define EMB_FORMAT_PEM 39
00064 #define EMB_FORMAT_PES 40
00065 #define EMB_FORMAT_PHB 41
00066 #define EMB_FORMAT_PHC 42
00067 #define EMB_FORMAT_PLT 43
00068 #define EMB_FORMAT_RGB 44
00069 #define EMB_FORMAT_SEW 45
00070 #define EMB_FORMAT_SHV 46
00071 #define EMB_FORMAT_SST 47
00072 #define EMB_FORMAT_STX 48
00073 #define EMB_FORMAT_SVG 49
00074 #define EMB_FORMAT_T01 50
00075 #define EMB_FORMAT_T09 51
00076 #define EMB_FORMAT_TAP 52
00077 #define EMB_FORMAT_THR 53
00078 #define EMB_FORMAT_TXT 54
00079 #define EMB_FORMAT_U00 55
00080 #define EMB_FORMAT_U01 56
00081 #define EMB_FORMAT_VIP 57
00082 #define EMB_FORMAT_VP3 58

```

```
00083 #define EMB_FORMAT_XXX      59
00084 #define EMB_FORMAT_ZSK      60
00085
00086 /* Thread color */
00087 #define Arc_Polyester      0
00088 #define Arc_Rayon          1
00089 #define CoatsAndClark_Rayon 2
00090 #define Exquisite_Polyester 3
00091 #define Fufu_Polyester      4
00092 #define Fufu_Rayon          5
00093 #define Hemingworth_Polyester 6
00094 #define Isacord_Polyester    7
00095 #define Isafil_Rayon        8
00096 #define Marathon_Polyester  9
00097 #define Marathon_Rayon      10
00098 #define Madeira_Polyester   11
00099 #define Madeira_Rayon       12
00100 #define Metro_Polyester    13
00101 #define Pantone            14
00102 #define RobisonAnton_Polyester 15
00103 #define RobisonAnton_Rayon  16
00104 #define Sigma_Polyester     17
00105 #define Sulky_Rayon         18
00106 #define ThreadArt_Rayon     19
00107 #define ThreadArt_Polyester  20
00108 #define ThreaDelight_Polyester 21
00109 #define Z102_Isacord_Polyester 22
00110 #define SVG_Colors           23
00111 #define hus_thread           24
00112 #define jef_thread           25
00113 #define pcm_thread           26
00114 #define pec_thread           27
00115 #define shv_thread           28
00116 #define dxf_color            29
00117
00118 #define EMB_ARRAY             0
00119 #define EMB_ARC               1
00120 #define EMB_CIRCLE             2
00121 #define EMB_DIM_DIAMETER      3
00122 #define EMB_DIM_LEADER         4
00123 #define EMB_ELLIPSE            5
00124 #define EMB_FLAG               6
00125 #define EMB_LINE               7
00126 #define EMB_IMAGE              8
00127 #define EMB_PATH               9
00128 #define EMB_POINT              10
00129 #define EMB_POLYGON             11
00130 #define EMB_POLYLINE            12
00131 #define EMB_RECT               13
00132 #define EMB_SPLINE              14
00133 #define EMB_STITCH              15
00134 #define EMB_TEXT_SINGLE         16
00135 #define EMB_TEXT_MULTI          17
00136 #define EMB_VECTOR              18
00137 #define EMB_THREAD              19
00138
00139 #define EMBFORMAT_UNSUPPORTED  0
00140 #define EMBFORMAT_STITCHONLY   1
00141 #define EMBFORMAT_OBJECTONLY   2
00142 #define EMBFORMAT_STCHANDOBJ  3 /* binary operation: 1+2=3 */
00143
00144 #define numberFormats          61
00145
00146 #define CHUNK_SIZE              128
00147
00148 #define EMB_MAX_LAYERS          10
00149 #define MAX_THREADS             256
00150 #define EMBFORMAT_MAXEXT        3
00151 /* maximum length of extension without dot */
00152 #define EMBFORMAT_MAXDESC       50
00153 /* the longest possible description string length */
00154 #define MAX_STITCHES           1000000
00155
00156
00157
00158 #if defined(_WIN32) && !defined(WIN32)
00159 #define WIN32
00160 #endif
00161
00162 /* When building a shared library,
00163 * use the proper export keyword depending on the compiler */
00164 #define EMB_PUBLIC
00165 #if defined(LIBEMBROIDERY_SHARED)
00166     #undef EMB_PUBLIC
00167     #if defined(__WIN32__)
00168         #define EMB_PUBLIC __declspec(dllexport)
00169     #else
```

```
00170     #define EMB_PUBLIC __attribute__ ((visibility("default")))
00171     #endif
00172 #endif
00173
00174 /* TYPEDEFS AND STRUCTS
00175 *****/
00176
00177 typedef float EmbReal;
00178
00179 typedef struct EmbColor_
00180 {
00181     unsigned char r;
00182     unsigned char g;
00183     unsigned char b;
00184 } EmbColor;
00185
00186 typedef struct EmbVector_
00187 {
00188     EmbReal x;
00189     EmbReal y;
00190 } EmbVector;
00191
00192 typedef struct EmbArray_ EmbArray;
00193
00194 typedef struct EmbImage_ {
00195     EmbVector position;
00196     EmbVector dimensions;
00197     unsigned char* data;
00198     int width;
00199     int height;
00200     char path[200];
00201     char name[200];
00202 } EmbImage;
00203
00204 typedef struct EmbBlock_ {
00205     EmbVector position;
00206 } EmbBlock;
00207
00208 typedef struct EmbAlignedDim_ {
00209     EmbVector position;
00210 } EmbAlignedDim;
00211
00212 typedef struct EmbAngularDim_ {
00213     EmbVector position;
00214 } EmbAngularDim;
00215
00216 typedef struct EmbArcLengthDim_ {
00217     EmbVector position;
00218 } EmbArcLengthDim;
00219
00220 typedef struct EmbDiameterDim_ {
00221     EmbVector position;
00222 } EmbDiameterDim;
00223
00224 typedef struct EmbLeaderDim_ {
00225     EmbVector position;
00226 } EmbLeaderDim;
00227
00228 typedef struct EmbLinearDim_ {
00229     EmbVector position;
00230 } EmbLinearDim;
00231
00232 typedef struct EmbOrdinateDim_ {
00233     EmbVector position;
00234 } EmbOrdinateDim;
00235
00236 typedef struct EmbRadiusDim_ {
00237     EmbVector position;
00238 } EmbRadiusDim;
00239
00240 typedef struct EmbInfiniteLine_ {
00241     EmbVector position;
00242 } EmbInfiniteLine;
00243
00244 typedef struct EmbRay_ {
00245     EmbVector position;
00246 } EmbRay;
00247
00248 typedef struct EmbTextMulti_ {
00249     EmbVector position;
00250 } EmbTextMulti;
00251
00252 typedef struct EmbTextSingle_ {
00253     EmbVector position;
00254 } EmbTextSingle;
00255
00256 typedef struct EmbTextMulti_ {
00257     EmbVector position;
00258 } EmbTextMulti;
00259
00260     EmbVector position;
00261     char text[200];
00262 } EmbTextMulti;
00263
00264 typedef struct EmbTextSingle_ {
00265     EmbVector position;
00266     char text[200];
00267 } EmbTextSingle;
```

```
00268
00269 typedef struct EmbTime_
00270 {
00271     unsigned int year;
00272     unsigned int month;
00273     unsigned int day;
00274     unsigned int hour;
00275     unsigned int minute;
00276     unsigned int second;
00277 } EmbTime;
00278
00279
00280 typedef struct EmbPoint_
00281 {
00282     EmbVector position;
00283     int lineType;
00284     EmbColor color;
00285 } EmbPoint;
00286
00287 typedef struct EmbLine_
00288 {
00289     EmbVector start;
00290     EmbVector end;
00291     int lineType;
00292     EmbColor color;
00293 } EmbLine;
00294
00295 typedef struct EmbPath_
00296 {
00297     EmbArray* pointList;
00298     EmbArray* flagList;
00299     int lineType;
00300     EmbColor color;
00301 } EmbPath;
00302
00303 typedef struct EmbStitch_
00304 {
00305     int flags; /* uses codes defined above */
00306     EmbReal x; /* absolute position (not relative) */
00307     EmbReal y; /* positive is up, units are in mm */
00308     int color; /* color number for this stitch */
00309     /* TODO: this should be called colorIndex since it is not an EmbColor */
00310 } EmbStitch;
00311
00312 typedef struct EmbThread_
00313 {
00314     EmbColor color;
00315     char description[50];
00316     char catalogNumber[30];
00317 } EmbThread;
00318
00319 typedef struct thread_color_ {
00320     char name[22];
00321     unsigned int hex_code;
00322     int manufacturer_code;
00323 } thread_color;
00324
00325 /* absolute position (not relative) */
00326 typedef struct EmbArc_
00327 {
00328     EmbVector start;
00329     EmbVector mid;
00330     EmbVector end;
00331 } EmbArc;
00332
00333 typedef struct EmbRect_
00334 {
00335     EmbReal top;
00336     EmbReal left;
00337     EmbReal bottom;
00338     EmbReal right;
00339     EmbReal rotation;
00340     EmbReal radius;
00341 } EmbRect;
00342
00343 typedef struct EmbCircle_
00344 {
00345     EmbVector center;
00346     EmbReal radius;
00347 } EmbCircle;
00348
00349 typedef EmbPath EmbPolygon;
00350 typedef EmbPath EmbPolyline;
00351 typedef int EmbFlag;
00352
00353 typedef struct EmbSatinOutline_
00354 {
```

```

00355     int length;
00356     EmbArray* side1;
00357     EmbArray* side2;
00358 } EmbSatinOutline;
00359
00360 typedef struct EmbEllipse_
00361 {
00362     EmbVector center;
00363     EmbVector radius;
00364     EmbReal rotation;
00365 } EmbEllipse;
00366
00367 typedef struct EmbBezier_ {
00368     EmbVector start;
00369     EmbVector control1;
00370     EmbVector control2;
00371     EmbVector end;
00372 } EmbBezier;
00373
00374 typedef struct EmbSpline_ {
00375     EmbArray *beziers;
00376 } EmbSpline;
00377
00378 typedef struct LSYSTEM {
00379     char axiom;
00380     char *alphabet;
00381     char *constants;
00382     char **rules;
00383 } L_system;
00384
00385 typedef struct EmbGeometry_ {
00386     union {
00387         EmbArc arc;
00388         EmbCircle circle;
00389         EmbColor color;
00390         EmbEllipse ellipse;
00391         EmbLine line;
00392         EmbPath path;
00393         EmbPoint point;
00394         EmbPolygon polygon;
00395         EmbPolyline polyline;
00396         EmbRect rect;
00397         EmbSpline spline;
00398         EmbVector vector;
00399     } object;
00400     EmbStitch stitch;
00401     EmbThread thread;
00402     int flag;
00403     int type;
00404     int lineType;
00405     EmbColor color;
00406 } EmbGeometry;
00407
00408 struct EmbArray_ {
00409     EmbGeometry *geometry;
00410     EmbStitch *stitch;
00411     EmbThread *thread;
00412     int count;
00413     int length;
00414     int type;
00415 };
00416
00417 typedef struct EmbLayer_
00418 {
00419     char name[100];
00420     EmbArray *geometry;
00421 } EmbLayer;
00422
00423 typedef struct EmbPattern_
00424 {
00425     unsigned int dstJumpsPerTrim;
00426     EmbVector home;
00427     EmbReal hoop_width;
00428     EmbReal hoop_height;
00429     EmbArray *thread_list;
00430     EmbArray *stitch_list;
00431     EmbArray *geometry;
00432     EmbLayer layer[EMB_MAX_LAYERS];
00433     int currentColorIndex;
00434 } EmbPattern;
00435
00436 typedef struct EmbFormatList_
00437 {
00438     char extension[2 + EMBFORMAT_MAXEXT];
00439     char description[EMBFORMAT_MAXDESC];
00440     char reader_state;
00441     char writer_state;

```

```

00442     int type;
00443     int color_only;
00444     int check_for_color_file;
00445     int write_external_color_file;
00446 } EmbFormatList;
00447
00448 /* Function Declarations
00449 ****
00450 EMB_PUBLIC int lindenmayer_system(L_system L, char* state, int iteration, int complete);
00451 EMB_PUBLIC int hilbert_curve(EmbPattern *pattern, int iterations);
00452
00453 EMB_PUBLIC int emb_identify_format(const char *ending);
00454 EMB_PUBLIC void testMain(int level);
00455 EMB_PUBLIC int convert(const char *inf, const char *outf);
00456
00457 EMB_PUBLIC EmbColor embColor_make(unsigned char r, unsigned char g, unsigned char b);
00458 EMB_PUBLIC EmbColor* embColor_create(unsigned char r, unsigned char g, unsigned char b);
00459 EMB_PUBLIC EmbColor embColor_fromHexStr(char* val);
00460 EMB_PUBLIC int embColor_distance(EmbColor a, EmbColor b);
00461
00462 EMB_PUBLIC EmbArray* embArray_create(int type);
00463 EMB_PUBLIC int embArray_resize(EmbArray *g);
00464 EMB_PUBLIC void embArray_copy(EmbArray *dst, EmbArray *src);
00465 EMB_PUBLIC int embArray_addArc(EmbArray* g, EmbArc arc);
00466 EMB_PUBLIC int embArray_addCircle(EmbArray* g, EmbCircle circle);
00467 EMB_PUBLIC int embArray_addEllipse(EmbArray* g, EmbEllipse ellipse);
00468 EMB_PUBLIC int embArray_addFlag(EmbArray* g, int flag);
00469 EMB_PUBLIC int embArray_addLine(EmbArray* g, EmbLine line);
00470 EMB_PUBLIC int embArray_addRect(EmbArray* g, EmbRect rect);
00471 EMB_PUBLIC int embArray_addPath(EmbArray* g, EmbPath p);
00472 EMB_PUBLIC int embArray_addPoint(EmbArray* g, EmbPoint p);
00473 EMB_PUBLIC int embArray_addPolygon(EmbArray* g, EmbPolygon p);
00474 EMB_PUBLIC int embArray_addPolyline(EmbArray* g, EmbPolyline p);
00475 /* EMB_PUBLIC int embArray_addSpline(EmbArray* g, EmbSpline p); */
00476 EMB_PUBLIC int embArray_addStitch(EmbArray* g, EmbStitch st);
00477 EMB_PUBLIC int embArray_addThread(EmbArray* g, EmbThread p);
00478 EMB_PUBLIC int embArray_addVector(EmbArray* g, EmbVector);
00479 EMB_PUBLIC void embArray_free(EmbArray* p);
00480
00481 EMB_PUBLIC EmbLine embLine_make(EmbReal x1, EmbReal y1, EmbReal x2, EmbReal y2);
00482
00483 EMB_PUBLIC void embLine_normalVector(EmbLine line, EmbVector* result, int clockwise);
00484 EMB_PUBLIC EmbVector embLine_intersectionPoint(EmbLine line1, EmbLine line2);
00485
00486 EMB_PUBLIC int embThread_findNearestColor(EmbColor color, EmbColor* colors, int n_colors);
00487 EMB_PUBLIC int embThread_findNearestThread(EmbColor color, EmbThread* threads, int n_threads);
00488 EMB_PUBLIC EmbThread embThread_getRandom(void);
00489
00490 EMB_PUBLIC void embVector_normalize(EmbVector vector, EmbVector* result);
00491 EMB_PUBLIC void embVector_multiply(EmbVector vector, EmbReal magnitude, EmbVector* result);
00492 EMB_PUBLIC EmbVector embVector_add(EmbVector v1, EmbVector v2);
00493 EMB_PUBLIC EmbVector embVector_average(EmbVector v1, EmbVector v2);
00494 EMB_PUBLIC EmbVector embVector_subtract(EmbVector v1, EmbVector v2);
00495 EMB_PUBLIC EmbReal embVector_dot(EmbVector v1, EmbVector v2);
00496 EMB_PUBLIC EmbReal embVector_cross(EmbVector v1, EmbVector v2);
00497 EMB_PUBLIC void embVector_transpose_product(EmbVector v1, EmbVector v2, EmbVector* result);
00498 EMB_PUBLIC EmbReal embVector_length(EmbVector vector);
00499 EMB_PUBLIC EmbReal embVector_relativeX(EmbVector a1, EmbVector a2, EmbVector a3);
00500 EMB_PUBLIC EmbReal embVector_relativeY(EmbVector a1, EmbVector a2, EmbVector a3);
00501 EMB_PUBLIC EmbReal embVector_angle(EmbVector v);
00502 EMB_PUBLIC EmbReal embVector_distance(EmbVector a, EmbVector b);
00503 EMB_PUBLIC EmbVector embVector_unit(EmbReal angle);
00504
00505 EMB_PUBLIC EmbArc embArc_init(void);
00506 EMB_PUBLIC char embArc_clockwise(EmbArc arc);
00507
00508 EMB_PUBLIC void getArcCenter(EmbArc arc, EmbVector *arcCenter);
00509 EMB_PUBLIC char getArcDataFromBulge(EmbReal bulge,
00510             EmbArc *arc,
00511             EmbReal* arcCenterX,           EmbReal* arcCenterY,
00512             EmbReal* radius,              EmbReal* diameter,
00513             EmbReal* chord,
00514             EmbReal* chordMidX,           EmbReal* chordMidY,
00515             EmbReal* sagitta,             EmbReal* apothem,
00516             EmbReal* incAngleInDegrees,   char* clockwise);
00517
00518 EMB_PUBLIC EmbCircle embCircle_init(void);
00519 EMB_PUBLIC int getCircleCircleIntersections(
00520     EmbCircle c0, EmbCircle c1, EmbVector *v0, EmbVector *v1);
00521 EMB_PUBLIC int getCircleTangentPoints(
00522     EmbCircle c, EmbVector p, EmbVector *v0, EmbVector *v1);
00523
00524 EMB_PUBLIC EmbEllipse embEllipse_init(void);
00525 EMB_PUBLIC EmbEllipse embEllipse_make(EmbReal cx, EmbReal cy, EmbReal rx, EmbReal ry);
00526 EMB_PUBLIC EmbReal embEllipse_diameterX(EmbEllipse ellipse);
00527 EMB_PUBLIC EmbReal embEllipse_diameterY(EmbEllipse ellipse);
00528 EMB_PUBLIC EmbReal embEllipse_width(EmbEllipse ellipse);

```

```

00529 EMB_PUBLIC EmbReal embEllipse_height(EmbEllipse ellipse);
00530 EMB_PUBLIC EmbReal embEllipse_area(EmbEllipse ellipse);
00531 EMB_PUBLIC EmbReal embEllipse_perimeter(EmbEllipse ellipse);
00532
00533 EMB_PUBLIC EmbImage embImage_create(int, int);
00534 EMB_PUBLIC void embImage_read(EmbImage *image, char *fname);
00535 EMB_PUBLIC int embImage_write(EmbImage *image, char *fname);
00536 EMB_PUBLIC void embImage_free(EmbImage *image);
00537
00538 EMB_PUBLIC EmbRect embRect_init(void);
00539 EMB_PUBLIC EmbReal embRect_area(EmbRect);
00540
00541 EMB_PUBLIC int threadColor(const char*, int brand);
00542 EMB_PUBLIC int threadColorNum(unsigned int color, int brand);
00543 EMB_PUBLIC const char* threadColorName(unsigned int color, int brand);
00544
00545 EMB_PUBLIC void embTime_initNow(EmbTime* t);
00546 EMB_PUBLIC EmbTime embTime_time(EmbTime* t);
00547
00548 EMB_PUBLIC void embSatinOutline_generateSatinOutline(EmbArray* lines, EmbReal thickness,
    EmbSatinOutline* result);
00549 EMB_PUBLIC EmbArray* embSatinOutline_renderStitches(EmbSatinOutline* result, EmbReal density);
00550
00551 EMB_PUBLIC EmbGeometry *embGeometry_init(int type_in);
00552 EMB_PUBLIC void embGeometry_free(EmbGeometry *obj);
00553 EMB_PUBLIC void embGeometry_move(EmbGeometry *obj, EmbVector delta);
00554 EMB_PUBLIC EmbRect embGeometry_boundingRect(EmbGeometry *obj);
00555 EMB_PUBLIC void embGeometry_vulcanize(EmbGeometry *obj);
00556
00557 EMB_PUBLIC EmbPattern* embPattern_create(void);
00558 EMB_PUBLIC void embPattern_hideStitchesOverLength(EmbPattern* p, int length);
00559 EMB_PUBLIC void embPattern_fixColorCount(EmbPattern* p);
00560 EMB_PUBLIC int embPattern_addThread(EmbPattern* p, EmbThread thread);
00561 EMB_PUBLIC void embPattern_addStitchAbs(EmbPattern* p, EmbReal x, EmbReal y, int flags, int
    isAutoColorIndex);
00562 EMB_PUBLIC void embPattern_addStitchRel(EmbPattern* p, EmbReal dx, EmbReal dy, int flags, int
    isAutoColorIndex);
00563 EMB_PUBLIC void embPattern_changeColor(EmbPattern* p, int index);
00564 EMB_PUBLIC void embPattern_free(EmbPattern* p);
00565 EMB_PUBLIC void embPattern_scale(EmbPattern* p, EmbReal scale);
00566 EMB_PUBLIC EmbReal embPattern_totalStitchLength(EmbPattern *pattern);
00567 EMB_PUBLIC EmbReal embPattern_minimumStitchLength(EmbPattern *pattern);
00568 EMB_PUBLIC EmbReal embPattern_maximumStitchLength(EmbPattern *pattern);
00569 EMB_PUBLIC void embPattern_lengthHistogram(EmbPattern *pattern, int *bin, int NUMBINS);
00570 EMB_PUBLIC int embPattern_realStitches(EmbPattern *pattern);
00571 EMB_PUBLIC int embPattern_jumpStitches(EmbPattern *pattern);
00572 EMB_PUBLIC int embPattern_trimStitches(EmbPattern *pattern);
00573 EMB_PUBLIC EmbRect embPattern_calcBoundingBox(EmbPattern* p);
00574 EMB_PUBLIC void embPattern_flipHorizontal(EmbPattern* p);
00575 EMB_PUBLIC void embPattern_flipVertical(EmbPattern* p);
00576 EMB_PUBLIC void embPattern_flip(EmbPattern* p, int horz, int vert);
00577 EMB_PUBLIC void embPattern_combineJumpStitches(EmbPattern* p);
00578 EMB_PUBLIC void embPattern_correctForMaxStitchLength(EmbPattern* p, EmbReal maxStitchLength, EmbReal
    maxJumpLength);
00579 EMB_PUBLIC void embPattern_center(EmbPattern* p);
00580 EMB_PUBLIC void embPattern_loadExternalColorFile(EmbPattern* p, const char* fileName);
00581 EMB_PUBLIC void embPattern_convertGeometry(EmbPattern* p);
00582 EMB_PUBLIC void embPattern_designDetails(EmbPattern *p);
00583 EMB_PUBLIC EmbPattern *embPattern_combine(EmbPattern *p1, EmbPattern *p2);
00584 EMB_PUBLIC int embPattern_color_count(EmbPattern *pattern, EmbColor startColor);
00585 EMB_PUBLIC void embPattern_end(EmbPattern* p);
00586 EMB_PUBLIC void embPattern_crossstitch(EmbPattern *pattern, EmbImage *, int threshhold);
00587 EMB_PUBLIC void embPattern_horizontal_fill(EmbPattern *pattern, EmbImage *, int threshhold);
00588 EMB_PUBLIC int embPattern_render(EmbPattern *pattern, char *fname);
00589 EMB_PUBLIC int embPattern_simulate(EmbPattern *pattern, char *fname);
00590
00591 EMB_PUBLIC void embPattern_addCircleAbs(EmbPattern* p, EmbCircle obj);
00592 EMB_PUBLIC void embPattern_addEllipseAbs(EmbPattern* p, EmbEllipse obj);
00593 EMB_PUBLIC void embPattern_addLineAbs(EmbPattern* p, EmbLine obj);
00594 EMB_PUBLIC void embPattern_addPathAbs(EmbPattern* p, EmbPath obj);
00595 EMB_PUBLIC void embPattern_addPointAbs(EmbPattern* p, EmbPoint obj);
00596 EMB_PUBLIC void embPattern_addPolygonAbs(EmbPattern* p, EmbPolygon obj);
00597 EMB_PUBLIC void embPattern_addPolylineAbs(EmbPattern* p, EmbPolyline obj);
00598 EMB_PUBLIC void embPattern_addRectAbs(EmbPattern* p, EmbRect obj);
00599
00600 EMB_PUBLIC void embPattern_copyStitchListToPolylines(EmbPattern* pattern);
00601 EMB_PUBLIC void embPattern_copyPolylinesToStitchList(EmbPattern* pattern);
00602 EMB_PUBLIC void embPattern_moveStitchListToPolylines(EmbPattern* pattern);
00603 EMB_PUBLIC void embPattern_movePolylinesToStitchList(EmbPattern* pattern);
00604
00605 EMB_PUBLIC char embPattern_read(EmbPattern *pattern, const char* fileName, int format);
00606 EMB_PUBLIC char embPattern_write(EmbPattern *pattern, const char* fileName, int format);
00607
00608 EMB_PUBLIC char embPattern_readAuto(EmbPattern *pattern, const char* fileName);
00609 EMB_PUBLIC char embPattern_writeAuto(EmbPattern *pattern, const char* fileName);
00610
00611 EMB_PUBLIC void report(int result, char *label);

```

```
00612 EMB_PUBLIC int full_test_matrix(char *fname);
00613
00614 EMB_PUBLIC int emb_round(EmbReal x);
00615 EMB_PUBLIC EmbReal radians(EmbReal degree);
00616 EMB_PUBLIC EmbReal degrees(EmbReal radian);
00617
00618 /* NON-MACRO CONSTANTS
00619 *****/
00620
00621 extern EmbFormatList formatTable[numberOfFormats];
00622 extern const int pecThreadCount;
00623 extern const int shvThreadCount;
00624 extern const EmbReal embConstantPi;
00625 extern const EmbThread husThreads[];
00626 extern const EmbThread jefThreads[];
00627 extern const EmbThread shvThreads[];
00628 extern const EmbThread pcmThreads[];
00629 extern const EmbThread pecThreads[];
00630 extern const unsigned char _dxfColorTable[][][3];
00631 extern EmbThread black_thread;
00632 extern const unsigned char vipDecodingTable[];
00633
00634 /* VARIABLES
00635 *****/
00636
00637 extern int emb_error;
00638 extern int emb_verbose;
00639
00640 #ifdef __cplusplus
00641 }
00642 #endif /* __cplusplus */
00643
00644 #endif /* LIBEMBROIDERY_HEADER */
```

4.7 src/embroidery_internal.h File Reference

```
#include "embroidery.h"
#include <stdio.h>
```

Data Structures

- struct [_bcf_file_difat](#)
- struct [_bcf_file_fat](#)
- struct [_bcf_directory_entry](#)
- struct [_bcf_directory](#)
- struct [_bcf_file_header](#)
- struct [_bcf_file](#)
- struct [_vp3Hoop](#)
- struct [ThredHeader_](#)
- struct [ThredExtension_](#)
- struct [SubDescriptor_](#)
- struct [StxThread_](#)
- struct [VipHeader_](#)
- struct [SvgAttribute_](#)
- struct [Huffman](#)
- struct [Compress](#)

Macros

- #define CompoundFileSector_MaxRegSector 0xFFFFFFFFA
- #define CompoundFileSector_DIFAT_Sector 0xFFFFFFFFC
- #define CompoundFileSector_FAT_Sector 0xFFFFFFFFD
- #define CompoundFileSector_EndOfChain 0xFFFFFFFFE
- #define CompoundFileSector_FreeSector 0xFFFFFFFFF
- #define ObjectTypeUnknown 0x00
- #define ObjectTypeStorage 0x01
- #define ObjectTypeStream 0x02
- #define ObjectTypeRootEntry 0x05
- #define CompoundFileStreamId_MaxRegularStreamId 0xFFFFFFFFA
- #define CompoundFileStreamId_NoStream 0xFFFFFFFFF
- #define ELEMENT_XML 0
- #define ELEMENT_A 1
- #define ELEMENT_ANIMATE 2
- #define ELEMENT_ANIMATECOLOR 3
- #define ELEMENT_ANIMATEMOTION 4
- #define ELEMENT_ANIMATETRANSFORM 5
- #define ELEMENT_ANIMATION 6
- #define ELEMENT_AUDIO 7
- #define ELEMENT_CIRCLE 8
- #define ELEMENT_DEFS 9
- #define ELEMENT_DESC 10
- #define ELEMENT_DISCARD 11
- #define ELEMENT_ELLIPSE 12
- #define ELEMENT_FONT 13
- #define ELEMENT_FONT_FACE 14
- #define ELEMENT_FONT_FACE_SRC 15
- #define ELEMENT_FONT_FACE_URI 16
- #define ELEMENT_FOREIGN_OBJECT 17
- #define ELEMENT_G 18
- #define ELEMENT_GLYPH 19
- #define ELEMENT_HANDLER 20
- #define ELEMENT_HKERN 21
- #define ELEMENT_IMAGE 22
- #define ELEMENT_LINE 23
- #define ELEMENT_LINEAR_GRADIENT 24
- #define ELEMENT_LISTENER 25
- #define ELEMENT_METADATA 26
- #define ELEMENT_MISSING_GLYPH 27
- #define ELEMENT_MPATH 28
- #define ELEMENT_PATH 29
- #define ELEMENT_POLYGON 30
- #define ELEMENT_POLYLINE 31
- #define ELEMENT_PREFETCH 32
- #define ELEMENT_RADIAL_GRADIENT 33
- #define ELEMENT_RECT 34
- #define ELEMENT_SCRIPT 35
- #define ELEMENT_SET 36
- #define ELEMENT_SOLID_COLOR 37
- #define ELEMENT_STOP 38
- #define ELEMENT_SVG 39
- #define ELEMENT_SWITCH 40
- #define ELEMENT_TBREAK 41

- #define ELEMENT_TEXT 42
- #define ELEMENT_TEXT_AREA 43
- #define ELEMENT_TITLE 44
- #define ELEMENT_TSPAN 45
- #define ELEMENT_USE 46
- #define ELEMENT_VIDEO 47
- #define RED_TERM_COLOR "\x1B[0;31m"
- #define GREEN_TERM_COLOR "\x1B[0;32m"
- #define YELLOW_TERM_COLOR "\x1B[1;33m"
- #define RESET_TERM_COLOR "\033[0m"
- #define HOOP_126X110 0
- #define HOOP_110X110 1
- #define HOOP_50X50 2
- #define HOOP_140X200 3
- #define HOOP_230X200 4
- #define EMB_MIN(A, B) (((A) < (B)) ? (A) : (B))
- #define EMB_MAX(A, B) (((A) > (B)) ? (A) : (B))
- #define EMB_BIG_ENDIAN 0
- #define EMB_LITTLE_ENDIAN 1
- #define ENDIAN_HOST EMB_LITTLE_ENDIAN
- #define EMB_INT16_BIG 2
- #define EMB_INT16_LITTLE 3
- #define EMB_INT32_BIG 4
- #define EMB_INT32_LITTLE 5
- #define PES0001 0
- #define PES0020 1
- #define PES0022 2
- #define PES0030 3
- #define PES0040 4
- #define PES0050 5
- #define PES0055 6
- #define PES0056 7
- #define PES0060 8
- #define PES0070 9
- #define PES0080 10
- #define PES0090 11
- #define PES0100 12
- #define N_PES VERSIONS 13
- #define DXF_VERSION_R10 "AC1006"
- #define DXF_VERSION_R11 "AC1009"
- #define DXF_VERSION_R12 "AC1009"
- #define DXF_VERSION_R13 "AC1012"
- #define DXF_VERSION_R14 "AC1014"
- #define DXF_VERSION_R15 "AC1015"
- #define DXF_VERSION_R18 "AC1018"
- #define DXF_VERSION_R21 "AC1021"
- #define DXF_VERSION_R24 "AC1024"
- #define DXF_VERSION_R27 "AC1027"
- #define DXF_VERSION_2000 "AC1015"
- #define DXF_VERSION_2002 "AC1015"
- #define DXF_VERSION_2004 "AC1018"
- #define DXF_VERSION_2006 "AC1018"
- #define DXF_VERSION_2007 "AC1021"
- #define DXF_VERSION_2009 "AC1021"
- #define DXF_VERSION_2010 "AC1024"

- #define DXF_VERSION_2013 "AC1027"
- #define SVG_CREATOR_NULL 0
- #define SVG_CREATOR_EMBROIDERMODDER 1
- #define SVG_CREATOR_ILLUSTRATOR 2
- #define SVG_CREATOR_INKSCAPE 3
- #define SVG_EXPECT_NULL 0
- #define SVG_EXPECT_ELEMENT 1
- #define SVG_EXPECT_ATTRIBUTE 2
- #define SVG_EXPECT_VALUE 3
- #define SVG_NULL 0
- #define SVG_ELEMENT 1
- #define SVG_PROPERTY 2
- #define SVG_MEDIA_PROPERTY 3
- #define SVG_ATTRIBUTE 4
- #define SVG_CATCH_ALL 5
- #define LINETO 0
- #define MOVETO 1
- #define BULGETOCONTROL 2
- #define BULGETOEND 4
- #define ELLIPSETORAD 8
- #define ELLIPSETOEND 16
- #define CUBICTOCONTROL1 32
- #define CUBICTOCONTROL2 64
- #define CUBICTOEND 128
- #define QUADTOCONTROL 256
- #define QUADTOEND 512

Typedefs

- typedef struct _bcf_file_difat bcf_file_difat
- typedef struct _bcf_file_fat bcf_file_fat
- typedef struct _bcf_directory_entry bcf_directory_entry
- typedef struct _bcf_directory bcf_directory
- typedef struct _bcf_file_header bcf_file_header
- typedef struct _bcf_file bcf_file
- typedef struct _vp3Hoop vp3Hoop
- typedef struct ThredHeader_ ThredHeader
- typedef struct ThredExtension_ ThredExtension
- typedef struct SubDescriptor_ SubDescriptor
- typedef struct StxThread_ StxThread
- typedef struct VipHeader_ VipHeader
- typedef struct SvgAttribute_ SvgAttribute
- typedef struct Huffman huffman
- typedef struct Compress compress

Enumerations

- enum CSV_EXPECT { CSV_EXPECT_NULL , CSV_EXPECT_QUOTE1 , CSV_EXPECT_QUOTE2 , CSV_EXPECT_COMMA }
- enum CSV_MODE {
 CSV_MODE_NULL , CSV_MODE_COMMENT , CSV_MODE_VARIABLE , CSV_MODE_THREAD ,
 CSV_MODE_STITCH }

Functions

- void `huffman_build_table (huffman *h)`
- int * `huffman_table_lookup (huffman *h, int byte_lookup, int *lengths)`
- int `compress_get_bits (compress *c, int length)`
- int `compress_pop (compress *c, int bit_count)`
- int `compress_read_variable_length (compress *c)`
- void `compress_load_character_length_huffman (compress *c)`
- void `compress_load_character_huffman (compress *c)`
- void `compress_load_distance_huffman (compress *c)`
- void `compress_load_block (compress *c)`
- int `compress_get_token (compress *c)`
- int `compress_get_position (compress *c)`
- void `readPecStitches (EmbPattern *pattern, FILE *file)`
- void `writePecStitches (EmbPattern *pattern, FILE *file, const char *filename)`
- int `decodeNewStitch (unsigned char value)`
- void `pfaffEncode (FILE *file, int x, int y, int flags)`
- `EmbReal pfaffDecode (unsigned char a1, unsigned char a2, unsigned char a3)`
- unsigned char `mitEncodeStitch (EmbReal value)`
- int `mitDecodeStitch (unsigned char value)`
- int `encode_tajima_ternary (unsigned char b[3], int x, int y)`
- void `decode_tajima_ternary (unsigned char b[3], int *x, int *y)`
- void `encode_t01_record (unsigned char b[3], int x, int y, int flags)`
- int `decode_t01_record (unsigned char b[3], int *x, int *y, int *flags)`
- void `readPESHeaderV5 (FILE *file, EmbPattern *pattern)`
- void `readPESHeaderV6 (FILE *file, EmbPattern *pattern)`
- void `readPESHeaderV7 (FILE *file, EmbPattern *pattern)`
- void `readPESHeaderV8 (FILE *file, EmbPattern *pattern)`
- void `readPESHeaderV9 (FILE *file, EmbPattern *pattern)`
- void `readPESHeaderV10 (FILE *file, EmbPattern *pattern)`
- void `readDescriptions (FILE *file, EmbPattern *pattern)`
- void `readHoopName (FILE *file, EmbPattern *pattern)`
- void `readImageString (FILE *file, EmbPattern *pattern)`
- void `readProgrammableFills (FILE *file, EmbPattern *pattern)`
- void `readMotifPatterns (FILE *file, EmbPattern *pattern)`
- void `readFeatherPatterns (FILE *file, EmbPattern *pattern)`
- void `readThreads (FILE *file, EmbPattern *pattern)`
- void `emblnt_read (FILE *f, char *label, void *b, int mode)`
- void `emblnt_write (FILE *f, char *label, void *b, int mode)`
- int `emb_readline (FILE *file, char *line, int maxLength)`
- int `bcfFile_read (FILE *file, bcf_file *bcfFile)`
- FILE * `GetFile (bcf_file *bcfFile, FILE *file, char *fileToFind)`
- void `bcf_file_free (bcf_file *bcfFile)`
- void `binaryReadString (FILE *file, char *buffer, int maxLength)`
- void `binaryReadUnicodeString (FILE *file, char *buffer, const int stringLength)`
- int `stringInArray (const char *s, const char **array)`
- void `fpad (FILE *f, char c, int n)`
- char * `copy_trim (char const *s)`
- char * `emb_optOut (EmbReal num, char *str)`
- void `write_24bit (FILE *file, int)`
- int `check_header_present (FILE *file, int minimum_header_length)`
- unsigned short `fread_uint16 (FILE *file)`
- short `fread_int16 (FILE *f)`
- void `safe_free (void *data)`
- void `binaryWriteUInt (FILE *f, unsigned int data)`

- `bcf_file_difat * bcf_difat_create (FILE *file, unsigned int fatSectors, const unsigned int sectorSize)`
- `unsigned int readFullSector (FILE *file, bcf_file_difat *bcfFile, unsigned int *numberOfDifatEntriesStillToRead)`
- `unsigned int numberOfEntriesInDifatSector (bcf_file_difat *fat)`
- `void bcf_file_difat_free (bcf_file_difat *difat)`
- `unsigned int entriesInDifatSector (bcf_file_difat *fat)`
- `bcf_file_fat * bcfFileFat_create (const unsigned int sectorSize)`
- `void loadFatFromSector (bcf_file_fat *fat, FILE *file)`
- `void bcf_file_fat_free (bcf_file_fat **fat)`
- `bcf_directory_entry * CompoundFileDirectoryEntry (FILE *file)`
- `bcf_directory * CompoundFileDirectory (const unsigned int maxNumberOfDirectoryEntries)`
- `void readNextSector (FILE *file, bcf_directory *dir)`
- `void bcf_directory_free (bcf_directory **dir)`
- `bcf_file_header bcfFileHeader_read (FILE *file)`
- `int bcfFileHeader_isValid (bcf_file_header header)`
- `int hus_compress (char *input, int size, char *output, int *out_size)`
- `int hus_decompress (char *input, int size, char *output, int *out_size)`
- `void testTangentPoints (EmbCircle c, EmbVector p, EmbVector *t0, EmbVector *t1)`
- `void printArcResults (EmbReal bulge, EmbArc arc, EmbReal centerX, EmbReal centerY, EmbReal radius, EmbReal diameter, EmbReal chord, EmbReal chordMidX, EmbReal chordMidY, EmbReal sagitta, EmbReal apothem, EmbReal incAngle, char clockwise)`
- `int create_test_file_1 (const char *outf)`
- `int create_test_file_2 (const char *outf)`
- `int create_test_file_3 (const char *outf)`
- `int testEmbCircle (void)`
- `int testEmbCircle_2 (void)`
- `int testGeomArc (void)`
- `int testThreadColor (void)`
- `int testEmbFormat (void)`
- `void embColor_read (FILE *f, EmbColor *c, int toRead)`
- `void embColor_write (FILE *f, EmbColor c, int toWrite)`
- `char read100 (EmbPattern *pattern, FILE *file)`
- `char write100 (EmbPattern *pattern, FILE *file)`
- `char read10o (EmbPattern *pattern, FILE *file)`
- `char write10o (EmbPattern *pattern, FILE *file)`
- `char readArt (EmbPattern *pattern, FILE *file)`
- `char writeArt (EmbPattern *pattern, FILE *file)`
- `char readBmc (EmbPattern *pattern, FILE *file)`
- `char writeBmc (EmbPattern *pattern, FILE *file)`
- `char readBro (EmbPattern *pattern, FILE *file)`
- `char writeBro (EmbPattern *pattern, FILE *file)`
- `char readCnd (EmbPattern *pattern, FILE *file)`
- `char writeCnd (EmbPattern *pattern, FILE *file)`
- `char readCol (EmbPattern *pattern, FILE *file)`
- `char writeCol (EmbPattern *pattern, FILE *file)`
- `char readCsd (EmbPattern *pattern, FILE *file)`
- `char writeCsd (EmbPattern *pattern, FILE *file)`
- `char readCsv (EmbPattern *pattern, FILE *file)`
- `char writeCsv (EmbPattern *pattern, FILE *file)`
- `char readDat (EmbPattern *pattern, FILE *file)`
- `char writeDat (EmbPattern *pattern, FILE *file)`
- `char readDem (EmbPattern *pattern, FILE *file)`
- `char writeDem (EmbPattern *pattern, FILE *file)`
- `char readDsb (EmbPattern *pattern, FILE *file)`
- `char writeDsb (EmbPattern *pattern, FILE *file)`

- char `readDst` (`EmbPattern` *pattern, `FILE` *file)
- char `writeDst` (`EmbPattern` *pattern, `FILE` *file)
- char `readDsz` (`EmbPattern` *pattern, `FILE` *file)
- char `writeDsz` (`EmbPattern` *pattern, `FILE` *file)
- char `readDxf` (`EmbPattern` *pattern, `FILE` *file)
- char `writeDxf` (`EmbPattern` *pattern, `FILE` *file)
- char `readEdr` (`EmbPattern` *pattern, `FILE` *file)
- char `writeEdr` (`EmbPattern` *pattern, `FILE` *file)
- char `readEmd` (`EmbPattern` *pattern, `FILE` *file)
- char `writeEmd` (`EmbPattern` *pattern, `FILE` *file)
- char `readExp` (`EmbPattern` *pattern, `FILE` *file)
- char `writeExp` (`EmbPattern` *pattern, `FILE` *file)
- char `readExy` (`EmbPattern` *pattern, `FILE` *file)
- char `writeExy` (`EmbPattern` *pattern, `FILE` *file)
- char `readEys` (`EmbPattern` *pattern, `FILE` *file)
- char `writeEys` (`EmbPattern` *pattern, `FILE` *file)
- char `readFxy` (`EmbPattern` *pattern, `FILE` *file)
- char `writeFxy` (`EmbPattern` *pattern, `FILE` *file)
- char `readGc` (`EmbPattern` *pattern, `FILE` *file)
- char `writeGc` (`EmbPattern` *pattern, `FILE` *file)
- char `readGnc` (`EmbPattern` *pattern, `FILE` *file)
- char `writeGnc` (`EmbPattern` *pattern, `FILE` *file)
- char `readGt` (`EmbPattern` *pattern, `FILE` *file)
- char `writeGt` (`EmbPattern` *pattern, `FILE` *file)
- char `readHus` (`EmbPattern` *pattern, `FILE` *file)
- char `writeHus` (`EmbPattern` *pattern, `FILE` *file)
- char `readInb` (`EmbPattern` *pattern, `FILE` *file)
- char `writeInb` (`EmbPattern` *pattern, `FILE` *file)
- char `readInf` (`EmbPattern` *pattern, `FILE` *file)
- char `writeInf` (`EmbPattern` *pattern, `FILE` *file)
- char `readJef` (`EmbPattern` *pattern, `FILE` *file)
- char `writeJef` (`EmbPattern` *pattern, `FILE` *file)
- char `readKsm` (`EmbPattern` *pattern, `FILE` *file)
- char `writeKsm` (`EmbPattern` *pattern, `FILE` *file)
- char `readMax` (`EmbPattern` *pattern, `FILE` *file)
- char `writeMax` (`EmbPattern` *pattern, `FILE` *file)
- char `readMit` (`EmbPattern` *pattern, `FILE` *file)
- char `writeMit` (`EmbPattern` *pattern, `FILE` *file)
- char `readNew` (`EmbPattern` *pattern, `FILE` *file)
- char `writeNew` (`EmbPattern` *pattern, `FILE` *file)
- char `readOfm` (`EmbPattern` *pattern, `FILE` *file)
- char `writeOfm` (`EmbPattern` *pattern, `FILE` *file)
- char `readPcd` (`EmbPattern` *pattern, `const char` *fileName, `FILE` *file)
- char `writePcd` (`EmbPattern` *pattern, `FILE` *file)
- char `readPcm` (`EmbPattern` *pattern, `FILE` *file)
- char `writePcm` (`EmbPattern` *pattern, `FILE` *file)
- char `readPcq` (`EmbPattern` *pattern, `const char` *fileName, `FILE` *file)
- char `writePcq` (`EmbPattern` *pattern, `FILE` *file)
- char `readPcs` (`EmbPattern` *pattern, `const char` *fileName, `FILE` *file)
- char `writePcs` (`EmbPattern` *pattern, `FILE` *file)
- char `readPec` (`EmbPattern` *pattern, `const char` *fileName, `FILE` *file)
- char `writePec` (`EmbPattern` *pattern, `const char` *fileName, `FILE` *file)
- char `readPel` (`EmbPattern` *pattern, `FILE` *file)
- char `writePel` (`EmbPattern` *pattern, `FILE` *file)
- char `readPem` (`EmbPattern` *pattern, `FILE` *file)

- char `writePem` (`EmbPattern` *pattern, `FILE` *file)
- char `readPes` (`EmbPattern` *pattern, const `char` *fileName, `FILE` *file)
- char `writePes` (`EmbPattern` *pattern, const `char` *fileName, `FILE` *file)
- char `readPhb` (`EmbPattern` *pattern, `FILE` *file)
- char `writePhb` (`EmbPattern` *pattern, `FILE` *file)
- char `readPhc` (`EmbPattern` *pattern, `FILE` *file)
- char `writePhc` (`EmbPattern` *pattern, `FILE` *file)
- char `readPlt` (`EmbPattern` *pattern, `FILE` *file)
- char `writePlt` (`EmbPattern` *pattern, `FILE` *file)
- char `readRgb` (`EmbPattern` *pattern, `FILE` *file)
- char `writeRgb` (`EmbPattern` *pattern, `FILE` *file)
- char `readSew` (`EmbPattern` *pattern, `FILE` *file)
- char `writeSew` (`EmbPattern` *pattern, `FILE` *file)
- char `readShv` (`EmbPattern` *pattern, `FILE` *file)
- char `writeShv` (`EmbPattern` *pattern, `FILE` *file)
- char `readSst` (`EmbPattern` *pattern, `FILE` *file)
- char `writeSst` (`EmbPattern` *pattern, `FILE` *file)
- char `readStx` (`EmbPattern` *pattern, `FILE` *file)
- char `writeStx` (`EmbPattern` *pattern, `FILE` *file)
- char `readSvg` (`EmbPattern` *pattern, `FILE` *file)
- char `writeSvg` (`EmbPattern` *pattern, `FILE` *file)
- char `readT01` (`EmbPattern` *pattern, `FILE` *file)
- char `writeT01` (`EmbPattern` *pattern, `FILE` *file)
- char `readT09` (`EmbPattern` *pattern, `FILE` *file)
- char `writeT09` (`EmbPattern` *pattern, `FILE` *file)
- char `readTap` (`EmbPattern` *pattern, `FILE` *file)
- char `writeTap` (`EmbPattern` *pattern, `FILE` *file)
- char `readThr` (`EmbPattern` *pattern, `FILE` *file)
- char `writeThr` (`EmbPattern` *pattern, `FILE` *file)
- char `readTxt` (`EmbPattern` *pattern, `FILE` *file)
- char `writeTxt` (`EmbPattern` *pattern, `FILE` *file)
- char `readU00` (`EmbPattern` *pattern, `FILE` *file)
- char `writeU00` (`EmbPattern` *pattern, `FILE` *file)
- char `readU01` (`EmbPattern` *pattern, `FILE` *file)
- char `writeU01` (`EmbPattern` *pattern, `FILE` *file)
- char `readVip` (`EmbPattern` *pattern, `FILE` *file)
- char `writeVip` (`EmbPattern` *pattern, `FILE` *file)
- char `readVp3` (`EmbPattern` *pattern, `FILE` *file)
- char `writeVp3` (`EmbPattern` *pattern, `FILE` *file)
- char `readXxx` (`EmbPattern` *pattern, `FILE` *file)
- char `writeXxx` (`EmbPattern` *pattern, `FILE` *file)
- char `readZsk` (`EmbPattern` *pattern, `FILE` *file)
- char `writeZsk` (`EmbPattern` *pattern, `FILE` *file)

Variables

- const `char` `imageWithFrame` [38][48]

4.7.1 Macro Definition Documentation

4.7.1.1 BULGETOCONTROL

```
#define BULGETOCONTROL 2
```

Definition at line 167 of file [embroidery_internal.h](#).

4.7.1.2 BULGETOEND

```
#define BULGETOEND 4
```

Definition at line 168 of file [embroidery_internal.h](#).

4.7.1.3 CompoundFileSector_DIFAT_Sector

```
#define CompoundFileSector_DIFAT_Sector 0xFFFFFFF0
```

Definition at line 13 of file [embroidery_internal.h](#).

4.7.1.4 CompoundFileSector_EndOfChain

```
#define CompoundFileSector_EndOfChain 0xFFFFFFFF
```

Definition at line 15 of file [embroidery_internal.h](#).

4.7.1.5 CompoundFileSector_FAT_Sector

```
#define CompoundFileSector_FAT_Sector 0xFFFFFFF0
```

Definition at line 14 of file [embroidery_internal.h](#).

4.7.1.6 CompoundFileSector_FreeSector

```
#define CompoundFileSector_FreeSector 0xFFFFFFFF
```

Definition at line 16 of file [embroidery_internal.h](#).

4.7.1.7 CompoundFileSector_MaxRegSector

```
#define CompoundFileSector_MaxRegSector 0xFFFFFFFFA
```

Type of sector

Definition at line 12 of file [embroidery_internal.h](#).

4.7.1.8 CompoundFileStreamId_MaxRegularStreamId

```
#define CompoundFileStreamId_MaxRegularStreamId 0xFFFFFFFFA
```

Special values for Stream Identifiers All real stream Ids are less than this

Definition at line 29 of file [embroidery_internal.h](#).

4.7.1.9 CompoundFileStreamId_NoStream

```
#define CompoundFileStreamId_NoStream 0xFFFFFFFFF
```

There is no valid stream Id

Definition at line 30 of file [embroidery_internal.h](#).

4.7.1.10 CUBICTOCONTROL1

```
#define CUBICTOCONTROL1 32
```

Definition at line 171 of file [embroidery_internal.h](#).

4.7.1.11 CUBICTOCONTROL2

```
#define CUBICTOCONTROL2 64
```

Definition at line 172 of file [embroidery_internal.h](#).

4.7.1.12 CUBICTOEND

```
#define CUBICTOEND 128
```

Definition at line 173 of file [embroidery_internal.h](#).

4.7.1.13 DXF_VERSION_2000

```
#define DXF_VERSION_2000 "AC1015"
```

Definition at line 135 of file [embroidery_internal.h](#).

4.7.1.14 DXF_VERSION_2002

```
#define DXF_VERSION_2002 "AC1015"
```

Definition at line 136 of file [embroidery_internal.h](#).

4.7.1.15 DXF_VERSION_2004

```
#define DXF_VERSION_2004 "AC1018"
```

Definition at line 137 of file [embroidery_internal.h](#).

4.7.1.16 DXF_VERSION_2006

```
#define DXF_VERSION_2006 "AC1018"
```

Definition at line 138 of file [embroidery_internal.h](#).

4.7.1.17 DXF_VERSION_2007

```
#define DXF_VERSION_2007 "AC1021"
```

Definition at line 139 of file [embroidery_internal.h](#).

4.7.1.18 DXF_VERSION_2009

```
#define DXF_VERSION_2009 "AC1021"
```

Definition at line 140 of file [embroidery_internal.h](#).

4.7.1.19 DXF_VERSION_2010

```
#define DXF_VERSION_2010 "AC1024"
```

Definition at line 141 of file [embroidery_internal.h](#).

4.7.1.20 DXF_VERSION_2013

```
#define DXF_VERSION_2013 "AC1027"
```

Definition at line 142 of file [embroidery_internal.h](#).

4.7.1.21 DXF_VERSION_R10

```
#define DXF_VERSION_R10 "AC1006"
```

Definition at line 124 of file [embroidery_internal.h](#).

4.7.1.22 DXF_VERSION_R11

```
#define DXF_VERSION_R11 "AC1009"
```

Definition at line 125 of file [embroidery_internal.h](#).

4.7.1.23 DXF_VERSION_R12

```
#define DXF_VERSION_R12 "AC1009"
```

Definition at line 126 of file [embroidery_internal.h](#).

4.7.1.24 DXF_VERSION_R13

```
#define DXF_VERSION_R13 "AC1012"
```

Definition at line 127 of file [embroidery_internal.h](#).

4.7.1.25 DXF_VERSION_R14

```
#define DXF_VERSION_R14 "AC1014"
```

Definition at line 128 of file [embroidery_internal.h](#).

4.7.1.26 DXF_VERSION_R15

```
#define DXF_VERSION_R15 "AC1015"
```

Definition at line 129 of file [embroidery_internal.h](#).

4.7.1.27 DXF_VERSION_R18

```
#define DXF_VERSION_R18 "AC1018"
```

Definition at line 130 of file [embroidery_internal.h](#).

4.7.1.28 DXF_VERSION_R21

```
#define DXF_VERSION_R21 "AC1021"
```

Definition at line 131 of file [embroidery_internal.h](#).

4.7.1.29 DXF_VERSION_R24

```
#define DXF_VERSION_R24 "AC1024"
```

Definition at line 132 of file [embroidery_internal.h](#).

4.7.1.30 DXF_VERSION_R27

```
#define DXF_VERSION_R27 "AC1027"
```

Definition at line 133 of file [embroidery_internal.h](#).

4.7.1.31 ELEMENT_A

```
#define ELEMENT_A 1
```

Definition at line 33 of file [embroidery_internal.h](#).

4.7.1.32 ELEMENT_ANIMATE

```
#define ELEMENT_ANIMATE 2
```

Definition at line 34 of file [embroidery_internal.h](#).

4.7.1.33 ELEMENT_ANIMATECOLOR

```
#define ELEMENT_ANIMATECOLOR 3
```

Definition at line 35 of file [embroidery_internal.h](#).

4.7.1.34 ELEMENT_ANIMATEMOTION

```
#define ELEMENT_ANIMATEMOTION 4
```

Definition at line 36 of file [embroidery_internal.h](#).

4.7.1.35 ELEMENT_ANIMATETRANSFORM

```
#define ELEMENT_ANIMATETRANSFORM 5
```

Definition at line 37 of file [embroidery_internal.h](#).

4.7.1.36 ELEMENT_ANIMATION

```
#define ELEMENT_ANIMATION 6
```

Definition at line 38 of file [embroidery_internal.h](#).

4.7.1.37 ELEMENT_AUDIO

```
#define ELEMENT_AUDIO 7
```

Definition at line 39 of file [embroidery_internal.h](#).

4.7.1.38 ELEMENT_CIRCLE

```
#define ELEMENT_CIRCLE 8
```

Definition at line 40 of file [embroidery_internal.h](#).

4.7.1.39 ELEMENT_DEFS

```
#define ELEMENT_DEFS 9
```

Definition at line 41 of file [embroidery_internal.h](#).

4.7.1.40 ELEMENT_DESC

```
#define ELEMENT_DESC 10
```

Definition at line 42 of file [embroidery_internal.h](#).

4.7.1.41 ELEMENT_DISCARD

```
#define ELEMENT_DISCARD 11
```

Definition at line 43 of file [embroidery_internal.h](#).

4.7.1.42 ELEMENT_ELLIPSE

```
#define ELEMENT_ELLIPSE 12
```

Definition at line [44](#) of file [embroidery_internal.h](#).

4.7.1.43 ELEMENT_FONT

```
#define ELEMENT_FONT 13
```

Definition at line [45](#) of file [embroidery_internal.h](#).

4.7.1.44 ELEMENT_FONT_FACE

```
#define ELEMENT_FONT_FACE 14
```

Definition at line [46](#) of file [embroidery_internal.h](#).

4.7.1.45 ELEMENT_FONT_FACE_SRC

```
#define ELEMENT_FONT_FACE_SRC 15
```

Definition at line [47](#) of file [embroidery_internal.h](#).

4.7.1.46 ELEMENT_FONT_FACE_URI

```
#define ELEMENT_FONT_FACE_URI 16
```

Definition at line [48](#) of file [embroidery_internal.h](#).

4.7.1.47 ELEMENT_FOREIGN_OBJECT

```
#define ELEMENT_FOREIGN_OBJECT 17
```

Definition at line [49](#) of file [embroidery_internal.h](#).

4.7.1.48 ELEMENT_G

```
#define ELEMENT_G 18
```

Definition at line 50 of file [embroidery_internal.h](#).

4.7.1.49 ELEMENT_GLYPH

```
#define ELEMENT_GLYPH 19
```

Definition at line 51 of file [embroidery_internal.h](#).

4.7.1.50 ELEMENT_HANDLER

```
#define ELEMENT_HANDLER 20
```

Definition at line 52 of file [embroidery_internal.h](#).

4.7.1.51 ELEMENT_HKERN

```
#define ELEMENT_HKERN 21
```

Definition at line 53 of file [embroidery_internal.h](#).

4.7.1.52 ELEMENT_IMAGE

```
#define ELEMENT_IMAGE 22
```

Definition at line 54 of file [embroidery_internal.h](#).

4.7.1.53 ELEMENT_LINE

```
#define ELEMENT_LINE 23
```

Definition at line 55 of file [embroidery_internal.h](#).

4.7.1.54 ELEMENT_LINEAR_GRADIENT

```
#define ELEMENT_LINEAR_GRADIENT 24
```

Definition at line 56 of file [embroidery_internal.h](#).

4.7.1.55 ELEMENT_LISTENER

```
#define ELEMENT_LISTENER 25
```

Definition at line 57 of file [embroidery_internal.h](#).

4.7.1.56 ELEMENT_METADATA

```
#define ELEMENT_METADATA 26
```

Definition at line 58 of file [embroidery_internal.h](#).

4.7.1.57 ELEMENT_MISSING_GLYPH

```
#define ELEMENT_MISSING_GLYPH 27
```

Definition at line 59 of file [embroidery_internal.h](#).

4.7.1.58 ELEMENT_MPATH

```
#define ELEMENT_MPATH 28
```

Definition at line 60 of file [embroidery_internal.h](#).

4.7.1.59 ELEMENT_PATH

```
#define ELEMENT_PATH 29
```

Definition at line 61 of file [embroidery_internal.h](#).

4.7.1.60 ELEMENT_Polygon

```
#define ELEMENT_Polygon 30
```

Definition at line [62](#) of file [embroidery_internal.h](#).

4.7.1.61 ELEMENT_Polyline

```
#define ELEMENT_Polyline 31
```

Definition at line [63](#) of file [embroidery_internal.h](#).

4.7.1.62 ELEMENT_PREFETCH

```
#define ELEMENT_PREFETCH 32
```

Definition at line [64](#) of file [embroidery_internal.h](#).

4.7.1.63 ELEMENT_RADIAL_GRADIENT

```
#define ELEMENT_RADIAL_GRADIENT 33
```

Definition at line [65](#) of file [embroidery_internal.h](#).

4.7.1.64 ELEMENT_RECT

```
#define ELEMENT_RECT 34
```

Definition at line [66](#) of file [embroidery_internal.h](#).

4.7.1.65 ELEMENT_SCRIPT

```
#define ELEMENT_SCRIPT 35
```

Definition at line [67](#) of file [embroidery_internal.h](#).

4.7.1.66 ELEMENT_SET

```
#define ELEMENT_SET 36
```

Definition at line [68](#) of file [embroidery_internal.h](#).

4.7.1.67 ELEMENT_SOLID_COLOR

```
#define ELEMENT_SOLID_COLOR 37
```

Definition at line [69](#) of file [embroidery_internal.h](#).

4.7.1.68 ELEMENT_STOP

```
#define ELEMENT_STOP 38
```

Definition at line [70](#) of file [embroidery_internal.h](#).

4.7.1.69 ELEMENT_SVG

```
#define ELEMENT_SVG 39
```

Definition at line [71](#) of file [embroidery_internal.h](#).

4.7.1.70 ELEMENT_SWITCH

```
#define ELEMENT_SWITCH 40
```

Definition at line [72](#) of file [embroidery_internal.h](#).

4.7.1.71 ELEMENT_TBREAK

```
#define ELEMENT_TBREAK 41
```

Definition at line [73](#) of file [embroidery_internal.h](#).

4.7.1.72 ELEMENT_TEXT

```
#define ELEMENT_TEXT 42
```

Definition at line 74 of file [embroidery_internal.h](#).

4.7.1.73 ELEMENT_TEXT_AREA

```
#define ELEMENT_TEXT_AREA 43
```

Definition at line 75 of file [embroidery_internal.h](#).

4.7.1.74 ELEMENT_TITLE

```
#define ELEMENT_TITLE 44
```

Definition at line 76 of file [embroidery_internal.h](#).

4.7.1.75 ELEMENT_TSPAN

```
#define ELEMENT_TSPAN 45
```

Definition at line 77 of file [embroidery_internal.h](#).

4.7.1.76 ELEMENT_USE

```
#define ELEMENT_USE 46
```

Definition at line 78 of file [embroidery_internal.h](#).

4.7.1.77 ELEMENT_VIDEO

```
#define ELEMENT_VIDEO 47
```

Definition at line 79 of file [embroidery_internal.h](#).

4.7.1.78 ELEMENT_XML

```
#define ELEMENT_XML 0
```

Definition at line 32 of file [embroidery_internal.h](#).

4.7.1.79 ELLIPSETOEND

```
#define ELLIPSETOEND 16
```

Definition at line 170 of file [embroidery_internal.h](#).

4.7.1.80 ELLIPSETORAD

```
#define ELLIPSETORAD 8
```

Definition at line 169 of file [embroidery_internal.h](#).

4.7.1.81 EMB_BIG_ENDIAN

```
#define EMB_BIG_ENDIAN 0
```

Definition at line 98 of file [embroidery_internal.h](#).

4.7.1.82 EMB_INT16_BIG

```
#define EMB_INT16_BIG 2
```

Definition at line 103 of file [embroidery_internal.h](#).

4.7.1.83 EMB_INT16_LITTLE

```
#define EMB_INT16_LITTLE 3
```

Definition at line 104 of file [embroidery_internal.h](#).

4.7.1.84 EMB_INT32_BIG

```
#define EMB_INT32_BIG 4
```

Definition at line 105 of file [embroidery_internal.h](#).

4.7.1.85 EMB_INT32_LITTLE

```
#define EMB_INT32_LITTLE 5
```

Definition at line 106 of file [embroidery_internal.h](#).

4.7.1.86 EMB_LITTLE_ENDIAN

```
#define EMB_LITTLE_ENDIAN 1
```

Definition at line 99 of file [embroidery_internal.h](#).

4.7.1.87 EMB_MAX

```
#define EMB_MAX(  
    A,  
    B ) (( (A) > (B) ) ? (A) : (B))
```

Definition at line 94 of file [embroidery_internal.h](#).

4.7.1.88 EMB_MIN

```
#define EMB_MIN(  
    A,  
    B ) (( (A) < (B) ) ? (A) : (B))
```

Definition at line 93 of file [embroidery_internal.h](#).

4.7.1.89 ENDIAN_HOST

```
#define ENDIAN_HOST EMB_LITTLE_ENDIAN
```

Definition at line 101 of file [embroidery_internal.h](#).

4.7.1.90 GREEN_TERM_COLOR

```
#define GREEN_TERM_COLOR "\x1B[0;32m"
```

Definition at line 83 of file [embroidery_internal.h](#).

4.7.1.91 HOOP_110X110

```
#define HOOP_110X110 1
```

Definition at line 88 of file [embroidery_internal.h](#).

4.7.1.92 HOOP_126X110

```
#define HOOP_126X110 0
```

Definition at line 87 of file [embroidery_internal.h](#).

4.7.1.93 HOOP_140X200

```
#define HOOP_140X200 3
```

Definition at line 90 of file [embroidery_internal.h](#).

4.7.1.94 HOOP_230X200

```
#define HOOP_230X200 4
```

Definition at line 91 of file [embroidery_internal.h](#).

4.7.1.95 HOOP_50X50

```
#define HOOP_50X50 2
```

Definition at line 89 of file [embroidery_internal.h](#).

4.7.1.96 LINETO

```
#define LINETO 0
```

Definition at line 165 of file [embroidery_internal.h](#).

4.7.1.97 MOVETO

```
#define MOVETO 1
```

Definition at line 166 of file [embroidery_internal.h](#).

4.7.1.98 N_PES VERSIONS

```
#define N_PES_VERSIONS 13
```

Definition at line 121 of file [embroidery_internal.h](#).

4.7.1.99 ObjectTypeRootEntry

```
#define ObjectTypeRootEntry 0x05
```

the root entry

Definition at line 24 of file [embroidery_internal.h](#).

4.7.1.100 ObjectTypeStorage

```
#define ObjectTypeStorage 0x01
```

a directory type object

Definition at line 22 of file [embroidery_internal.h](#).

4.7.1.101 ObjectTypeStream

```
#define ObjectTypeStream 0x02
```

a file type object

Definition at line [23](#) of file [embroidery_internal.h](#).

4.7.1.102 ObjectTypeUnknown

```
#define ObjectTypeUnknown 0x00
```

Type of directory object Probably unallocated

Definition at line [21](#) of file [embroidery_internal.h](#).

4.7.1.103 PES0001

```
#define PES0001 0
```

Definition at line [108](#) of file [embroidery_internal.h](#).

4.7.1.104 PES0020

```
#define PES0020 1
```

Definition at line [109](#) of file [embroidery_internal.h](#).

4.7.1.105 PES0022

```
#define PES0022 2
```

Definition at line [110](#) of file [embroidery_internal.h](#).

4.7.1.106 PES0030

```
#define PES0030 3
```

Definition at line 111 of file [embroidery_internal.h](#).

4.7.1.107 PES0040

```
#define PES0040 4
```

Definition at line 112 of file [embroidery_internal.h](#).

4.7.1.108 PES0050

```
#define PES0050 5
```

Definition at line 113 of file [embroidery_internal.h](#).

4.7.1.109 PES0055

```
#define PES0055 6
```

Definition at line 114 of file [embroidery_internal.h](#).

4.7.1.110 PES0056

```
#define PES0056 7
```

Definition at line 115 of file [embroidery_internal.h](#).

4.7.1.111 PES0060

```
#define PES0060 8
```

Definition at line 116 of file [embroidery_internal.h](#).

4.7.1.112 PES0070

```
#define PES0070 9
```

Definition at line 117 of file [embroidery_internal.h](#).

4.7.1.113 PES0080

```
#define PES0080 10
```

Definition at line 118 of file [embroidery_internal.h](#).

4.7.1.114 PES0090

```
#define PES0090 11
```

Definition at line 119 of file [embroidery_internal.h](#).

4.7.1.115 PES0100

```
#define PES0100 12
```

Definition at line 120 of file [embroidery_internal.h](#).

4.7.1.116 QUADTOCONTROL

```
#define QUADTOCONTROL 256
```

Definition at line 174 of file [embroidery_internal.h](#).

4.7.1.117 QUADTOEND

```
#define QUADTOEND 512
```

Definition at line 175 of file [embroidery_internal.h](#).

4.7.1.118 RED_TERM_COLOR

```
#define RED_TERM_COLOR "\x1B[0;31m"
```

Definition at line 82 of file [embroidery_internal.h](#).

4.7.1.119 RESET_TERM_COLOR

```
#define RESET_TERM_COLOR "\033[0m"
```

Definition at line 85 of file [embroidery_internal.h](#).

4.7.1.120 SVG_ATTRIBUTE

```
#define SVG_ATTRIBUTE 4
```

Definition at line 161 of file [embroidery_internal.h](#).

4.7.1.121 SVG_CATCH_ALL

```
#define SVG_CATCH_ALL 5
```

Definition at line 162 of file [embroidery_internal.h](#).

4.7.1.122 SVG_CREATOR_EMBROIDERMODDER

```
#define SVG_CREATOR_EMBROIDERMODDER 1
```

Definition at line 145 of file [embroidery_internal.h](#).

4.7.1.123 SVG_CREATOR_ILLUSTRATOR

```
#define SVG_CREATOR_ILLUSTRATOR 2
```

Definition at line 146 of file [embroidery_internal.h](#).

4.7.1.124 **SVG_CREATOR_INKSCAPE**

```
#define SVG_CREATOR_INKSCAPE 3
```

Definition at line 147 of file [embroidery_internal.h](#).

4.7.1.125 **SVG_CREATOR_NULL**

```
#define SVG_CREATOR_NULL 0
```

Definition at line 144 of file [embroidery_internal.h](#).

4.7.1.126 **SVG_ELEMENT**

```
#define SVG_ELEMENT 1
```

Definition at line 158 of file [embroidery_internal.h](#).

4.7.1.127 **SVG_EXPECT_ATTRIBUTE**

```
#define SVG_EXPECT_ATTRIBUTE 2
```

Definition at line 151 of file [embroidery_internal.h](#).

4.7.1.128 **SVG_EXPECT_ELEMENT**

```
#define SVG_EXPECT_ELEMENT 1
```

Definition at line 150 of file [embroidery_internal.h](#).

4.7.1.129 **SVG_EXPECT_NULL**

```
#define SVG_EXPECT_NULL 0
```

Definition at line 149 of file [embroidery_internal.h](#).

4.7.1.130 SVG_EXPECT_VALUE

```
#define SVG_EXPECT_VALUE 3
```

Definition at line 152 of file [embroidery_internal.h](#).

4.7.1.131 SVG_MEDIA_PROPERTY

```
#define SVG_MEDIA_PROPERTY 3
```

Definition at line 160 of file [embroidery_internal.h](#).

4.7.1.132 SVG_NULL

```
#define SVG_NULL 0
```

Definition at line 157 of file [embroidery_internal.h](#).

4.7.1.133 SVG_PROPERTY

```
#define SVG_PROPERTY 2
```

Definition at line 159 of file [embroidery_internal.h](#).

4.7.1.134 YELLOW_TERM_COLOR

```
#define YELLOW_TERM_COLOR "\x1B[1;33m"
```

Definition at line 84 of file [embroidery_internal.h](#).

4.7.2 Typedef Documentation

4.7.2.1 bcf_directory

```
typedef struct _bcf_directory bcf_directory
```

4.7.2.2 **bcf_directory_entry**

```
typedef struct _bcf_directory_entry bcf_directory_entry
```

4.7.2.3 **bcf_file**

```
typedef struct _bcf_file bcf_file
```

4.7.2.4 **bcf_file_difat**

```
typedef struct _bcf_file_difat bcf_file_difat
```

4.7.2.5 **bcf_file_fat**

```
typedef struct _bcf_file_fat bcf_file_fat
```

4.7.2.6 **bcf_file_header**

```
typedef struct _bcf_file_header bcf_file_header
```

4.7.2.7 **compress**

```
typedef struct Compress compress
```

4.7.2.8 **huffman**

```
typedef struct Huffman huffman
```

4.7.2.9 **StxThread**

```
typedef struct StxThread_ StxThread
```

4.7.2.10 SubDescriptor

```
typedef struct SubDescriptor_ SubDescriptor
```

4.7.2.11 SvgAttribute

```
typedef struct SvgAttribute_ SvgAttribute
```

4.7.2.12 ThredExtension

```
typedef struct ThredExtension_ ThredExtension
```

4.7.2.13 ThredHeader

```
typedef struct ThredHeader_ ThredHeader
```

4.7.2.14 VipHeader

```
typedef struct VipHeader_ VipHeader
```

4.7.2.15 vp3Hoop

```
typedef struct _vp3Hoop vp3Hoop
```

4.7.3 Enumeration Type Documentation

4.7.3.1 CSV_EXPECT

```
enum CSV_EXPECT
```

Enumerator

CSV_EXPECT_NULL	
CSV_EXPECT_QUOTE1	
CSV_EXPECT_QUOTE2	
CSV_EXPECT_COMMA	

Definition at line 336 of file [embroidery_internal.h](#).

4.7.3.2 CSV_MODE

```
enum CSV_MODE
```

Enumerator

CSV_MODE_NULL	
CSV_MODE_COMMENT	
CSV_MODE_VARIABLE	
CSV_MODE_THREAD	
CSV_MODE_STITCH	

Definition at line 344 of file [embroidery_internal.h](#).

4.7.4 Function Documentation

4.7.4.1 bcf_difat_create()

```
bcf_file_difat * bcf_difat_create (
    FILE * file,
    unsigned int fatSectors,
    const unsigned int sectorSize )
```

Definition at line 276 of file [main.c](#).

4.7.4.2 bcf_directory_free()

```
void bcf_directory_free (
    bcf_directory ** dir )
```

Definition at line 448 of file [main.c](#).

4.7.4.3 bcf_file_difat_free()

```
void bcf_file_difat_free (
    bcf_file_difat * difat )
```

4.7.4.4 bcf_file_fat_free()

```
void bcf_file_fat_free (
    bcf_file_fat ** fat )
```

4.7.4.5 bcf_file_free()

```
void bcf_file_free (
    bcf_file * bcfFile )
```

Definition at line 267 of file [main.c](#).

4.7.4.6 bcfFile_read()

```
int bcfFile_read (
    FILE * file,
    bcf_file * bcfFile )
```

Definition at line 196 of file [main.c](#).

4.7.4.7 bcfFileFat_create()

```
bcf_file_fat * bcfFileFat_create (
    const unsigned int sectorSize )
```

Definition at line 467 of file [main.c](#).

4.7.4.8 bcfFileHeader_isValid()

```
int bcfFileHeader_isValid (
    bcf_file_header header )
```

4.7.4.9 **bcfFileHeader_read()**

```
bcf_file_header bcfFileHeader_read (
    FILE * file )
```

Definition at line [495](#) of file [main.c](#).

4.7.4.10 **binaryReadString()**

```
void binaryReadString (
    FILE * file,
    char * buffer,
    int maxLength )
```

Definition at line [766](#) of file [main.c](#).

4.7.4.11 **binaryReadUnicodeString()**

```
void binaryReadUnicodeString (
    FILE * file,
    char * buffer,
    const int stringLength )
```

Definition at line [777](#) of file [main.c](#).

4.7.4.12 **binaryWriteUInt()**

```
void binaryWriteUInt (
    FILE * f,
    unsigned int data )
```

Definition at line [247](#) of file [formats.c](#).

4.7.4.13 **check_header_present()**

```
int check_header_present (
    FILE * file,
    int minimum_header_length )
```

Definition at line [149](#) of file [main.c](#).

4.7.4.14 CompoundFileDirectory()

```
bcf_directory * CompoundFileDirectory (
    const unsigned int maxNumberOfDirectoryEntries )
```

Definition at line 360 of file [main.c](#).

4.7.4.15 CompoundFileDirectoryEntry()

```
bcf_directory_entry * CompoundFileDirectoryEntry (
    FILE * file )
```

Definition at line 390 of file [main.c](#).

4.7.4.16 compress_get_bits()

```
int compress_get_bits (
    compress * c,
    int length )
```

Definition at line 106 of file [compress.c](#).

4.7.4.17 compress_get_position()

```
int compress_get_position (
    compress * c )
```

Definition at line 251 of file [compress.c](#).

4.7.4.18 compress_get_token()

```
int compress_get_token (
    compress * c )
```

Definition at line 239 of file [compress.c](#).

4.7.4.19 compress_load_block()

```
void compress_load_block (
    compress * c )
```

Definition at line [231](#) of file [compress.c](#).

4.7.4.20 compress_load_character_huffman()

```
void compress_load_character_huffman (
    compress * c )
```

Definition at line [184](#) of file [compress.c](#).

4.7.4.21 compress_load_character_length_huffman()

```
void compress_load_character_length_huffman (
    compress * c )
```

Definition at line [162](#) of file [compress.c](#).

4.7.4.22 compress_load_distance_huffman()

```
void compress_load_distance_huffman (
    compress * c )
```

Definition at line [215](#) of file [compress.c](#).

4.7.4.23 compress_pop()

```
int compress_pop (
    compress * c,
    int bit_count )
```

Definition at line [131](#) of file [compress.c](#).

4.7.4.24 compress_read_variable_length()

```
int compress_read_variable_length (
    compress * c )
```

Definition at line 143 of file [compress.c](#).

4.7.4.25 copy_trim()

```
char * copy_trim (
    char const * s )
```

Definition at line 851 of file [main.c](#).

4.7.4.26 create_test_file_1()

```
int create_test_file_1 (
    const char * outf )
```

4.7.4.27 create_test_file_2()

```
int create_test_file_2 (
    const char * outf )
```

4.7.4.28 create_test_file_3()

```
int create_test_file_3 (
    const char * outf )
```

4.7.4.29 decode_t01_record()

```
int decode_t01_record (
    unsigned char b[3],
    int * x,
    int * y,
    int * flags )
```

Definition at line 74 of file [encoding.c](#).

4.7.4.30 decode_tajima_ternary()

```
void decode_tajima_ternary (
    unsigned char b[3],
    int * x,
    int * y )
```

Definition at line [229](#) of file [encoding.c](#).

4.7.4.31 decodeNewStitch()

```
int decodeNewStitch (
    unsigned char value )
```

Definition at line [337](#) of file [encoding.c](#).

4.7.4.32 emb_optOut()

```
char * emb_optOut (
    EmbReal num,
    char * str )
```

Optimizes the number (*num*) for output to a text file and returns it as a string (*str*).

Definition at line [868](#) of file [main.c](#).

4.7.4.33 emb_readline()

```
int emb_readline (
    FILE * file,
    char * line,
    int maxLength )
```

Definition at line [809](#) of file [main.c](#).

4.7.4.34 embColor_read()

```
void embColor_read (
    FILE * f,
    EmbColor * c,
    int toRead )
```

Definition at line [681](#) of file [main.c](#).

4.7.4.35 embColor_write()

```
void embColor_write (
    FILE * f,
    EmbColor c,
    int toWrite )
```

Definition at line [691](#) of file [main.c](#).

4.7.4.36 embInt_read()

```
void embInt_read (
    FILE * f,
    char * label,
    void * b,
    int mode )
```

Definition at line [350](#) of file [encoding.c](#).

4.7.4.37 embInt_write()

```
void embInt_write (
    FILE * f,
    char * label,
    void * b,
    int mode )
```

Definition at line [381](#) of file [encoding.c](#).

4.7.4.38 encode_t01_record()

```
void encode_t01_record (
    unsigned char b[3],
    int x,
    int y,
    int flags )
```

Definition at line [99](#) of file [encoding.c](#).

4.7.4.39 encode_tajima_ternary()

```
int encode_tajima_ternary (
    unsigned char b[3],
    int x,
    int y )
```

Definition at line 118 of file [encoding.c](#).

4.7.4.40 entriesInDifatSector()

```
unsigned int entriesInDifatSector (
    bcf_file_difat * fat )
```

Definition at line 308 of file [main.c](#).

4.7.4.41 fpad()

```
void fpad (
    FILE * f,
    char c,
    int n )
```

Definition at line 207 of file [formats.c](#).

4.7.4.42 fread_int16()

```
short fread_int16 (
    FILE * f )
```

Definition at line 183 of file [formats.c](#).

4.7.4.43 fread_uint16()

```
unsigned short fread_uint16 (
    FILE * file )
```

Definition at line 191 of file [formats.c](#).

4.7.4.44 GetFile()

```
FILE * GetFile (
    bcf_file * bcfFile,
    FILE * file,
    char * fileToFind )
```

Definition at line [233](#) of file [main.c](#).

4.7.4.45 huffman_build_table()

```
void huffman_build_table (
    huffman * h )
```

Definition at line [60](#) of file [compress.c](#).

4.7.4.46 huffman_table_lookup()

```
int * huffman_table_lookup (
    huffman * h,
    int byte_lookup,
    int * lengths )
```

4.7.4.47 hus_compress()

```
int hus_compress (
    char * input,
    int size,
    char * output,
    int * out_size )
```

Definition at line [44](#) of file [compress.c](#).

4.7.4.48 hus_decompress()

```
int hus_decompress (
    char * input,
    int size,
    char * output,
    int * out_size )
```

Definition at line [264](#) of file [compress.c](#).

4.7.4.49 **loadFatFromSector()**

```
void loadFatFromSector (
    bcf_file_fat * fat,
    FILE * file )
```

Definition at line [481](#) of file [main.c](#).

4.7.4.50 **mitDecodeStitch()**

```
int mitDecodeStitch (
    unsigned char value )
```

Definition at line [330](#) of file [encoding.c](#).

4.7.4.51 **mitEncodeStitch()**

```
unsigned char mitEncodeStitch (
    EmbReal value )
```

Definition at line [323](#) of file [encoding.c](#).

4.7.4.52 **numberOfEntriesInDifatSector()**

```
unsigned int numberOfEntriesInDifatSector (
    bcf_file_difat * fat )
```

4.7.4.53 **pfaffDecode()**

```
EmbReal pfaffDecode (
    unsigned char a1,
    unsigned char a2,
    unsigned char a3 )
```

Definition at line [315](#) of file [encoding.c](#).

4.7.4.54 pfaffEncode()

```
void pfaffEncode (
    FILE * file,
    int x,
    int y,
    int flags )
```

Definition at line 295 of file [encoding.c](#).

4.7.4.55 printArcResults()

```
void printArcResults (
    EmbReal bulge,
    EmbArc arc,
    EmbReal centerX,
    EmbReal centerY,
    EmbReal radius,
    EmbReal diameter,
    EmbReal chord,
    EmbReal chordMidX,
    EmbReal chordMidY,
    EmbReal sagitta,
    EmbReal apothem,
    EmbReal incAngle,
    char clockwise )
```

4.7.4.56 read100()

```
char read100 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.57 read10o()

```
char read10o (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.58 readArt()

```
char readArt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.59 `readBmc()`

```
char readBmc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.60 `readBro()`

```
char readBro (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.61 `readCnd()`

```
char readCnd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.62 `readCol()`

```
char readCol (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.63 `readCsd()`

```
char readCsd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.64 `readCsv()`

```
char readCsv (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.65 readDat()

```
char readDat (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.66 readDem()

```
char readDem (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.67 readDescriptions()

```
void readDescriptions (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.68 readDsb()

```
char readDsb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.69 readDst()

```
char readDst (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.70 readDsz()

```
char readDsz (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.71 `readDxf()`

```
char readDxf (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.72 `readEdr()`

```
char readEdr (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.73 `readEmd()`

```
char readEmd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.74 `readExp()`

```
char readExp (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.75 `readExy()`

```
char readExy (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.76 `readEys()`

```
char readEys (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.77 readFeatherPatterns()

```
void readFeatherPatterns (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.78 readFullSector()

```
unsigned int readFullSector (
    FILE * file,
    bcf_file_difat * bcfFile,
    unsigned int * numberOfDifatEntriesStillToRead )
```

Definition at line 314 of file [main.c](#).

4.7.4.79 readFxy()

```
char readFxy (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.80 readGc()

```
char readGc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.81 readGnc()

```
char readGnc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.82 readGt()

```
char readGt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.83 readHoopName()

```
void readHoopName (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.84 readHus()

```
char readHus (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.85 readImageString()

```
void readImageString (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.86 readInb()

```
char readInb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.87 readInf()

```
char readInf (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.88 readJef()

```
char readJef (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.89 readKsm()

```
char readKsm (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.90 readMax()

```
char readMax (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.91 readMit()

```
char readMit (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.92 readMotifPatterns()

```
void readMotifPatterns (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.93 readNew()

```
char readNew (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.94 readNextSector()

```
void readNextSector (
    FILE * file,
    bcf_directory * dir )
```

Definition at line [426](#) of file [main.c](#).

4.7.4.95 readOfm()

```
char readOfm (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.96 readPcd()

```
char readPcd (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.97 readPcm()

```
char readPcm (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.98 readPcq()

```
char readPcq (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.99 readPcs()

```
char readPcs (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.100 readPec()

```
char readPec (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.101 readPecStitches()

```
void readPecStitches (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.102 readPel()

```
char readPel (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.103 readPem()

```
char readPem (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.104 readPes()

```
char readPes (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.105 readPESHeaderV10()

```
void readPESHeaderV10 (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.106 readPESHeaderV5()

```
void readPESHeaderV5 (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.107 readPESHeaderV6()

```
void readPESHeaderV6 (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.108 readPESHeaderV7()

```
void readPESHeaderV7 (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.109 readPESHeaderV8()

```
void readPESHeaderV8 (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.110 readPESHeaderV9()

```
void readPESHeaderV9 (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.111 readPhb()

```
char readPhb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.112 readPhc()

```
char readPhc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.113 readPlt()

```
char readPlt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.114 readProgrammableFills()

```
void readProgrammableFills (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.115 readRgb()

```
char readRgb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.116 readSew()

```
char readSew (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.117 readShv()

```
char readShv (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.118 readSst()

```
char readSst (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.119 `readStx()`

```
char readStx (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.120 `readSvg()`

```
char readSvg (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.121 `readT01()`

```
char readT01 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.122 `readT09()`

```
char readT09 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.123 `readTap()`

```
char readTap (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.124 `readThr()`

```
char readThr (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.125 readThreads()

```
void readThreads (
    FILE * file,
    EmbPattern * pattern )
```

4.7.4.126 readTxt()

```
char readTxt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.127 readU00()

```
char readU00 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.128 readU01()

```
char readU01 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.129 readVip()

```
char readVip (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.130 readVp3()

```
char readVp3 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.131 `readXxx()`

```
char readXxx (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.132 `readZsk()`

```
char readZsk (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.133 `safe_free()`

```
void safe_free (
    void * data )
```

Definition at line 129 of file [formats.c](#).

4.7.4.134 `stringInArray()`

```
int stringInArray (
    const char * s,
    const char ** array )
```

Tests for the presence of a string *s* in the supplied *array*.

The end of the array is marked by an empty string.

Returns

0 if not present 1 if present.

Definition at line 797 of file [main.c](#).

4.7.4.135 `testEmbCircle()`

```
int testEmbCircle (
    void )
```

4.7.4.136 testEmbCircle_2()

```
int testEmbCircle_2 (
    void )
```

4.7.4.137 testEmbFormat()

```
int testEmbFormat (
    void )
```

4.7.4.138 testGeomArc()

```
int testGeomArc (
    void )
```

4.7.4.139 testTangentPoints()

```
void testTangentPoints (
    EmbCircle c,
    EmbVector p,
    EmbVector * t0,
    EmbVector * t1 )
```

4.7.4.140 testThreadColor()

```
int testThreadColor (
    void )
```

4.7.4.141 write100()

```
char write100 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.142 write10o()

```
char write10o (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.143 write_24bit()

```
void write_24bit (
    FILE * file,
    int x )
```

Definition at line [660](#) of file [main.c](#).

4.7.4.144 writeArt()

```
char writeArt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.145 writeBmc()

```
char writeBmc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.146 writeBro()

```
char writeBro (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.147 writeCnd()

```
char writeCnd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.148 writeCol()

```
char writeCol (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.149 writeCsd()

```
char writeCsd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.150 writeCsv()

```
char writeCsv (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.151 writeDat()

```
char writeDat (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.152 writeDem()

```
char writeDem (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.153 writeDsb()

```
char writeDsb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.154 writeDst()

```
char writeDst (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.155 writeDsz()

```
char writeDsz (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.156 writeDxf()

```
char writeDxf (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.157 writeEdr()

```
char writeEdr (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.158 writeEmd()

```
char writeEmd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.159 writeExp()

```
char writeExp (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.160 writeExy()

```
char writeExy (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.161 writeEys()

```
char writeEys (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.162 writeFxy()

```
char writeFxy (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.163 writeGc()

```
char writeGc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.164 writeGnc()

```
char writeGnc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.165 writeGt()

```
char writeGt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.166 writeHus()

```
char writeHus (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.167 writelnb()

```
char writelnb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.168 writelnf()

```
char writelnf (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.169 writeJef()

```
char writeJef (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.170 writeKsm()

```
char writeKsm (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.171 writeMax()

```
char writeMax (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.172 writeMit()

```
char writeMit (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.173 writeNew()

```
char writeNew (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.174 writeOfm()

```
char writeOfm (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.175 writePcd()

```
char writePcd (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.176 writePcm()

```
char writePcm (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.177 writePcq()

```
char writePcq (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.178 writePcs()

```
char writePcs (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.179 writePec()

```
char writePec (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.180 writePecStitches()

```
void writePecStitches (
    EmbPattern * pattern,
    FILE * file,
    const char * filename )
```

4.7.4.181 writePel()

```
char writePel (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.182 writePem()

```
char writePem (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.183 writePes()

```
char writePes (
    EmbPattern * pattern,
    const char * fileName,
    FILE * file )
```

4.7.4.184 writePhb()

```
char writePhb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.185 writePhc()

```
char writePhc (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.186 writePlt()

```
char writePlt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.187 writeRgb()

```
char writeRgb (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.188 writeSew()

```
char writeSew (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.189 writeShv()

```
char writeShv (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.190 writeSst()

```
char writeSst (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.191 writeStx()

```
char writeStx (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.192 writeSvg()

```
char writeSvg (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.193 writeT01()

```
char writeT01 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.194 writeT09()

```
char writeT09 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.195 writeTap()

```
char writeTap (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.196 writeThr()

```
char writeThr (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.197 writeTxt()

```
char writeTxt (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.198 writeU00()

```
char writeU00 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.199 writeU01()

```
char writeU01 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.200 writeVip()

```
char writeVip (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.201 writeVp3()

```
char writeVp3 (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.202 writeXxx()

```
char writeXxx (
    EmbPattern * pattern,
    FILE * file )
```

4.7.4.203 writeZsk()

```
char writeZsk (
    EmbPattern * pattern,
    FILE * file )
```

4.7.5 Variable Documentation

4.7.5.1 imageWithFrame

```
const char imageWithFrame[38][48] [extern]
```

Definition at line 87 of file [formats.c](#).

4.8 embroidery_internal.h

[Go to the documentation of this file.](#)

```
00001 #ifndef LIBEMBROIDERY_INTERNAL_HEADER_
00002 #define LIBEMBROIDERY_INTERNAL_HEADER_
00003
00004 #include "embroidery.h"
00005
00006 /* For FILE */
00007 #include <stdio.h>
00008
00012 #define CompoundFileSector_MaxRegSector 0xFFFFFFF8
00013 #define CompoundFileSector_DIFAT_Sector 0xFFFFFFF8
00014 #define CompoundFileSector_FAT_Sector 0xFFFFFFF8
00015 #define CompoundFileSector_EndOfChain 0xFFFFFFF8
00016 #define CompoundFileSector_FreeSector 0xFFFFFFF8
00017
00021 #define ObjectTypeUnknown 0x00
00022 #define ObjectTypeStorage 0x01
00023 #define ObjectTypeStream 0x02
00024 #define ObjectTypeRootEntry 0x05
00029 #define CompoundFileStreamId_MaxRegularStreamId 0xFFFFFFF8
00030 #define CompoundFileStreamId_NoStream 0xFFFFFFF8
00032 #define ELEMENT_XML 0
00033 #define ELEMENT_A 1
00034 #define ELEMENT_ANIMATE 2
00035 #define ELEMENT_ANIMATECOLOR 3
00036 #define ELEMENT_ANIMATEMOTION 4
00037 #define ELEMENT_ANIMATETRANSFORM 5
00038 #define ELEMENT_ANIMATION 6
00039 #define ELEMENT_AUDIO 7
00040 #define ELEMENT_CIRCLE 8
00041 #define ELEMENT_DEFS 9
00042 #define ELEMENT_DESC 10
00043 #define ELEMENT_DISCARD 11
00044 #define ELEMENT_ELLIPSE 12
00045 #define ELEMENT_FONT 13
```

```
00046 #define ELEMENT_FONT_FACE      14
00047 #define ELEMENT_FONT_FACE_SRC   15
00048 #define ELEMENT_FONT_FACE_URI   16
00049 #define ELEMENT_FOREIGN_OBJECT  17
00050 #define ELEMENT_G             18
00051 #define ELEMENT_GLYPH        19
00052 #define ELEMENT_HANDLER     20
00053 #define ELEMENT_HKERN       21
00054 #define ELEMENT_IMAGE        22
00055 #define ELEMENT_LINE         23
00056 #define ELEMENT_LINEAR_GRADIENT 24
00057 #define ELEMENT_LISTENER     25
00058 #define ELEMENT_METADATA    26
00059 #define ELEMENT_MISSING_GLYPH 27
00060 #define ELEMENT_MPATH        28
00061 #define ELEMENT_PATH         29
00062 #define ELEMENT_POLYGON      30
00063 #define ELEMENT_POLYLINE     31
00064 #define ELEMENT_PREFETCH    32
00065 #define ELEMENT_RADIAL_GRADIENT 33
00066 #define ELEMENT_RECT        34
00067 #define ELEMENT_SCRIPT      35
00068 #define ELEMENT_SET          36
00069 #define ELEMENT_SOLID_COLOR 37
00070 #define ELEMENT_STOP        38
00071 #define ELEMENT_SVG          39
00072 #define ELEMENT_SWITCH      40
00073 #define ELEMENT_TBREAK      41
00074 #define ELEMENT_TEXT        42
00075 #define ELEMENT_TEXT_AREA   43
00076 #define ELEMENT_TITLE       44
00077 #define ELEMENT_TSPAN        45
00078 #define ELEMENT_USE         46
00079 #define ELEMENT_VIDEO      47
00080
00081 /* INTERNAL DEFINES */
00082 #define RED_TERM_COLOR      "\x1B[0;31m"
00083 #define GREEN_TERM_COLOR    "\x1B[0;32m"
00084 #define YELLOW_TERM_COLOR   "\x1B[1;33m"
00085 #define RESET_TERM_COLOR   "\033[0m"
00086
00087 #define HOOP_126X110        0
00088 #define HOOP_110X110        1
00089 #define HOOP_50X50          2
00090 #define HOOP_140X200        3
00091 #define HOOP_230X200        4
00092
00093 #define EMB_MIN(A, B) (((A) < (B)) ? (A) : (B))
00094 #define EMB_MAX(A, B) (((A) > (B)) ? (A) : (B))
00095
00096 /* Libembroidery's handling of integer types.
00097 */
00098 #define EMB_BIG_ENDIAN      0
00099 #define EMB_LITTLE_ENDIAN   1
00100
00101 #define ENDIAN_HOST          EMB_LITTLE_ENDIAN
00102
00103 #define EMB_INT16_BIG        2
00104 #define EMB_INT16_LITTLE     3
00105 #define EMB_INT32_BIG        4
00106 #define EMB_INT32_LITTLE     5
00107
00108 #define PES0001        0
00109 #define PES0020        1
00110 #define PES0022        2
00111 #define PES0030        3
00112 #define PES0040        4
00113 #define PES0050        5
00114 #define PES0055        6
00115 #define PES0056        7
00116 #define PES0060        8
00117 #define PES0070        9
00118 #define PES0080       10
00119 #define PES0090       11
00120 #define PES0100       12
00121 #define N_PES_VERSIONS 13
00122
00123 /* DXF Version Identifiers */
00124 #define DXF_VERSION_R10 "AC1006"
00125 #define DXF_VERSION_R11 "AC1009"
00126 #define DXF_VERSION_R12 "AC1009"
00127 #define DXF_VERSION_R13 "AC1012"
00128 #define DXF_VERSION_R14 "AC1014"
00129 #define DXF_VERSION_R15 "AC1015"
00130 #define DXF_VERSION_R18 "AC1018"
00131 #define DXF_VERSION_R21 "AC1021"
00132 #define DXF_VERSION_R24 "AC1024"
```

```

00133 #define DXF_VERSION_R27 "AC1027"
00134
00135 #define DXF_VERSION_2000 "AC1015"
00136 #define DXF_VERSION_2002 "AC1015"
00137 #define DXF_VERSION_2004 "AC1018"
00138 #define DXF_VERSION_2006 "AC1018"
00139 #define DXF_VERSION_2007 "AC1021"
00140 #define DXF_VERSION_2009 "AC1021"
00141 #define DXF_VERSION_2010 "AC1024"
00142 #define DXF_VERSION_2013 "AC1027"
00143
00144 #define SVG_CREATOR_NULL 0
00145 #define SVG_CREATOR_EMBROIDERMODDER 1
00146 #define SVG_CREATOR_ILLUSTRATOR 2
00147 #define SVG_CREATOR_INKSCAPE 3
00148
00149 #define SVG_EXPECT_NULL 0
00150 #define SVG_EXPECT_ELEMENT 1
00151 #define SVG_EXPECT_ATTRIBUTE 2
00152 #define SVG_EXPECT_VALUE 3
00153
00154 /* SVG_TYPES
00155 * -----
00156 */
00157 #define SVG_NULL 0
00158 #define SVG_ELEMENT 1
00159 #define SVG_PROPERTY 2
00160 #define SVG_MEDIA_PROPERTY 3
00161 #define SVG_ATTRIBUTE 4
00162 #define SVG_CATCH_ALL 5
00163
00164 /* path flag codes */
00165 #define LINETO 0
00166 #define MOVETO 1
00167 #define BULGETOCONTROL 2
00168 #define BULGETOEND 4
00169 #define ELLIPSETORAD 8
00170 #define ELLIPSETOEND 16
00171 #define CUBICTOCONTROL1 32
00172 #define CUBICTOCONTROL2 64
00173 #define CUBICTOEND 128
00174 #define QUADTOCONTROL 256
00175 #define QUADTOEND 512
00176
00177 /* STRUCTS
00178 *****/
00179
00180 /* double-indirection file allocation table references */
00181 typedef struct _bcf_file_difat
00182 {
00183     unsigned int fatSectorCount;
00184     unsigned int fatSectorEntries[109];
00185     unsigned int sectorSize;
00186 } bcf_file_difat;
00187
00188 typedef struct _bcf_file_fat
00189 {
00190     int fatEntryCount;
00191     unsigned int fatEntries[255]; /* maybe make this dynamic */
00192     unsigned int numberOfEntriesInFatSector;
00193 } bcf_file_fat;
00194
00195 typedef struct _bcf_directory_entry
00196 {
00197     char directoryEntryName[32];
00198     unsigned short directoryEntryNameLength;
00199     unsigned char objectType;
00200     unsigned char colorFlag;
00201     unsigned int leftSiblingId;
00202     unsigned int rightSiblingId;
00203     unsigned int childId;
00204     unsigned char CLSID[16];
00205     unsigned int stateBits;
00206     EmbTime creationTime;
00207     EmbTime modifiedTime;
00208     unsigned int startingSectorLocation;
00209     unsigned long streamSize; /* should be long long but in our case we shouldn't need
00210     it, and hard to support on c89 cross platform */
00211     unsigned int streamSizeHigh; /* store the high int of streamsize */
00212     struct _bcf_directory_entry* next;
00213 }
00214 typedef struct _bcf_directory
00215 {
00216     bcf_directory_entry* dirEntries;
00217     unsigned int maxNumberOfDirectoryEntries;
00218     /* TODO: possibly add a directory tree in the future */

```

```

00219
00220 } bcf_directory;
00221
00222 typedef struct _bcf_file_header
00223 {
00224     unsigned char signature[8];
00225     unsigned char CLSID[16]; /* TODO: this should be a separate type */
00226     unsigned short minorVersion;
00227     unsigned short majorVersion;
00228     unsigned short byteOrder;
00229     unsigned short sectorShift;
00230     unsigned short miniSectorShift;
00231     unsigned short reserved1;
00232     unsigned int reserved2;
00233     unsigned int numberOfDirectorySectors;
00234     unsigned int numberOfFATSectors;
00235     unsigned int firstDirectorySectorLocation;
00236     unsigned int transactionSignatureNumber;
00237     unsigned int miniStreamCutoffSize;
00238     unsigned int firstMiniFATSectorLocation;
00239     unsigned int numberOfMiniFatSectors;
00240     unsigned int firstDifatSectorLocation;
00241     unsigned int numberOfDifatSectors;
00242 } bcf_file_header;
00243
00244 typedef struct _bcf_file
00245 {
00246     bcf_file_header header;
00247     bcf_file_difat* difat;
00248     bcf_file_fat* fat;
00249     bcf_directory* directory;
00250 } bcf_file;
00251
00252 typedef struct _vp3Hoop
00253 {
00254     int right;
00255     int bottom;
00256     int left;
00257     int top;
00258     int threadLength;
00259     char unknown2;
00260     unsigned char numberOfColors;
00261     unsigned short unknown3;
00262     int unknown4;
00263     int numberOfBytesRemaining;
00264
00265     int xOffset;
00266     int yOffset;
00267
00268     unsigned char byte1;
00269     unsigned char byte2;
00270     unsigned char byte3;
00271
00272     /* Centered hoop dimensions */
00273     int right2;
00274     int left2;
00275     int bottom2;
00276     int top2;
00277
00278     int width;
00279     int height;
00280 } vp3Hoop;
00281
00282 typedef struct ThredHeader_ /* thred file header */
00283 {
00284     unsigned int sigVersion; /* signature and version */
00285     unsigned int length; /* length of ThredHeader + length of stitch data */
00286     unsigned short numStitches; /* number of stitches */
00287     unsigned short hoopSize; /* size of hoop */
00288     unsigned short reserved[7]; /* reserved for expansion */
00289 } ThredHeader;
00290
00291 typedef struct ThredExtension_ /* thred v1.0 file header extension */
00292 {
00293     float hoopX; /* hoop size x dimension in 1/6 mm units */
00294     float hoopY; /* hoop size y dimension in 1/6 mm units */
00295     float stitchGranularity; /* stitches per millimeter--not implemented */
00296     char creatorName[50]; /* name of the file creator */
00297     char modifierName[50]; /* name of last file modifier */
00298     char auxFormat; /* auxiliary file format, 0=PCS,1=DST,2=PES */
00299     char reserved[31]; /* reserved for expansion */
00300 } ThredExtension;
00301
00302 typedef struct SubDescriptor_
00303 {
00304     int someNum; /* TODO: better variable naming */
00305     int someInt; /* TODO: better variable naming */

```

```
00306     int someOtherInt; /* TODO: better variable naming */
00307     char* colorCode;
00308     char* colorName;
00309 } SubDescriptor;
00310
00311 typedef struct StxThread_
00312 {
00313     char* colorCode;
00314     char* colorName;
00315     char* sectionName;
00316     SubDescriptor* subDescriptors;
00317     EmbColor stxColor;
00318 } StxThread;
00319
00320 typedef struct VipHeader_ {
00321     int magicCode;
00322     int numberofStitches;
00323     int numberofColors;
00324     short positiveXHoopSize;
00325     short positiveYHoopSize;
00326     short negativeXHoopSize;
00327     short negativeYHoopSize;
00328     int attributeOffset;
00329     int xOffset;
00330     int yOffset;
00331     unsigned char stringVal[8];
00332     short unknown;
00333     int colorLength;
00334 } VipHeader;
00335
00336 typedef enum
00337 {
00338     CSV_EXPECT_NULL,
00339     CSV_EXPECT_QUOTE1,
00340     CSV_EXPECT_QUOTE2,
00341     CSV_EXPECT_COMMA
00342 } CSV_EXPECT;
00343
00344 typedef enum
00345 {
00346     CSV_MODE_NULL,
00347     CSV_MODE_COMMENT,
00348     CSV_MODE_VARIABLE,
00349     CSV_MODE_THREAD,
00350     CSV_MODE_STITCH
00351 } CSV_MODE;
00352
00353 typedef struct SvgAttribute_
00354 {
00355     char* name;
00356     char* value;
00357 } SvgAttribute;
00358
00359 typedef struct Huffman {
00360     int default_value;
00361     int lengths[1000];
00362     int nlenghts;
00363     int table[1000];
00364     int table_width;
00365     int ntable;
00366 } huffman;
00367
00368 typedef struct Compress {
00369     int bit_position;
00370     char *input_data;
00371     int input_length;
00372     int bits_total;
00373     int block_elements;
00374     huffman character_length_huffman;
00375     huffman character_huffman;
00376     huffman distance_huffman;
00377 } compress;
00378
00379 /* Function Declarations
00380 *****/
00381 void huffman_build_table(huffman *h);
00382 int *huffman_table_lookup(huffman *h, int byte_lookup, int *lengths);
00383
00384 int compress_get_bits(compress *c, int length);
00385 int compress_pop(compress *c, int bit_count);
00386 int compress_read_variable_length(compress *c);
00387 void compress_load_character_length_huffman(compress *c);
00388 void compress_load_character_huffman(compress *c);
00389 void compress_load_distance_huffman(compress *c);
00390 void compress_load_block(compress *c);
00391 int compress_get_token(compress *c);
00392 int compress_get_position(compress *c);
```

```

00393
00394 void readPecStitches(EmbPattern* pattern, FILE* file);
00395 void writePecStitches(EmbPattern* pattern, FILE* file, const char* filename);
00396
00397 int decodeNewStitch(unsigned char value);
00398
00399 void pfaffEncode(FILE* file, int x, int y, int flags);
00400 EmbReal pfaffDecode(unsigned char a1, unsigned char a2, unsigned char a3);
00401
00402 unsigned char mitEncodeStitch(EmbReal value);
00403 int mitDecodeStitch(unsigned char value);
00404
00405 int encode_tajima_ternary(unsigned char b[3], int x, int y);
00406 void decode_tajima_ternary(unsigned char b[3], int *x, int *y);
00407
00408 void encode_t01_record(unsigned char b[3], int x, int y, int flags);
00409 int decode_t01_record(unsigned char b[3], int *x, int *y, int *flags);
00410 void readPESHeaderV5(FILE* file, EmbPattern* pattern);
00411 void readPESHeaderV6(FILE* file, EmbPattern* pattern);
00412 void readPESHeaderV7(FILE* file, EmbPattern* pattern);
00413 void readPESHeaderV8(FILE* file, EmbPattern* pattern);
00414 void readPESHeaderV9(FILE* file, EmbPattern* pattern);
00415 void readPESHeaderV10(FILE* file, EmbPattern* pattern);
00416
00417 void readDescriptions(FILE* file, EmbPattern* pattern);
00418 void readHoopName(FILE* file, EmbPattern* pattern);
00419 void readImageString(FILE* file, EmbPattern* pattern);
00420 void readProgrammableFills(FILE* file, EmbPattern* pattern);
00421 void readMotifPatterns(FILE* file, EmbPattern* pattern);
00422 void readFeatherPatterns(FILE* file, EmbPattern* pattern);
00423 void readThreads(FILE* file, EmbPattern* pattern);
00424
00425 void embInt_read(FILE* f, char *label, void *b, int mode);
00426 void embInt_write(FILE* f, char *label, void *b, int mode);
00427 int emb_readline(FILE* file, char *line, int maxLength);
00428
00429 int bcfFile_read(FILE* file, bcf_file* bcfFile);
00430 FILE* GetFile(bcf_file* bcfFile, FILE* file, char* fileToFind);
00431 void bcf_file_free(bcf_file* bcfFile);
00432
00433 void binaryReadString(FILE* file, char *buffer, int maxLength);
00434 void binaryReadUnicodeString(FILE* file, char *buffer, const int stringLength);
00435
00436 int stringInArray(const char *s, const char **array);
00437 void fpad(FILE *f, char c, int n);
00438 char *copy_trim(char const *s);
00439 char* emb_optOut(EmbReal num, char* str);
00440
00441 void write_24bit(FILE* file, int);
00442 int check_header_present(FILE* file, int minimum_header_length);
00443
00444 unsigned short fread_uint16(FILE *file);
00445 short fread_int16(FILE* f);
00446 void safe_free(void *data);
00447 void embInt_read(FILE* f, char *label, void *b, int mode);
00448
00449 void binaryWriteUInt(FILE* f, unsigned int data);
00450
00451 bcf_file_difat* bcf_difat_create(FILE* file, unsigned int fatSectors, const unsigned int sectorSize);
00452 unsigned int readFullSector(FILE* file, bcf_file_difat* bcfFile, unsigned int* numberDifatEntriesStillToRead);
00453 unsigned int numberEntriesInDifatSector(bcf_file_difat* fat);
00454 void bcf_file_difat_free(bcf_file_difat* difat);
00455
00456 unsigned int entriesInDifatSector(bcf_file_difat* fat);
00457 bcf_file_fat* bcfFileFat_create(const unsigned int sectorSize);
00458 void loadFatFromSector(bcf_file_fat* fat, FILE* file);
00459 void bcf_file_fat_free(bcf_file_fat** fat);
00460
00461 bcf_directory_entry* CompoundFileDirectoryEntry(FILE* file);
00462 bcf_directory* CompoundFileDirectory(const unsigned int maxNumberOfDirectoryEntries);
00463 void readNextSector(FILE* file, bcf_directory* dir);
00464 void bcf_directory_free(bcf_directory** dir);
00465
00466 bcf_file_header bcfFileHeader_read(FILE* file);
00467 int bcfFileHeader_isValid(bcf_file_header header);
00468
00469 int hus_compress(char* input, int size, char* output, int *out_size);
00470 int hus_decompress(char* input, int size, char* output, int *out_size);
00471
00472 int encode_tajima_ternary(unsigned char b[3], int x, int y);
00473 void decode_tajima_ternary(unsigned char b[3], int *x, int *y);
00474 void testTangentPoints(EmbCircle c, EmbVector p, EmbVector *t0, EmbVector *t1);
00475 void printArcResults(EmbReal bulge, EmbArc arc,
00476                         EmbReal centerX,   EmbReal centerY,
00477                         EmbReal radius,    EmbReal diameter,
00478                         EmbReal chord,
```

```

00479             EmbReal chordMidX, EmbReal chordMidY,
00480             EmbReal sagitta,   EmbReal apothem,
00481             EmbReal incAngle, char    clockwise);
00482 int  create_test_file_1(const char* outfit);
00483 int  create_test_file_2(const char* outfit);
00484 int  create_test_file_3(const char* outfit);
00485 int  testEmbCircle(void);
00486 int  testEmbCircle_2(void);
00487 int  testGeomArc(void);
00488 int  testThreadColor(void);
00489 int  testEmbFormat(void);
00490
00491 void embColor_read(FILE *f, EmbColor *c, int toRead);
00492 void embColor_write(FILE *f, EmbColor c, int toWrite);
00493
00494 char  read100(EmbPattern *pattern, FILE* file);
00495 char  write100(EmbPattern *pattern, FILE* file);
00496 char  read10o(EmbPattern *pattern, FILE* file);
00497 char  write10o(EmbPattern *pattern, FILE* file);
00498 char  readArt(EmbPattern *pattern, FILE* file);
00499 char  writeArt(EmbPattern *pattern, FILE* file);
00500 char  readBmc(EmbPattern *pattern, FILE* file);
00501 char  writeBmc(EmbPattern *pattern, FILE* file);
00502 char  readBro(EmbPattern *pattern, FILE* file);
00503 char  writeBro(EmbPattern *pattern, FILE* file);
00504 char  readCnd(EmbPattern *pattern, FILE* file);
00505 char  writeCnd(EmbPattern *pattern, FILE* file);
00506 char  readCol(EmbPattern *pattern, FILE* file);
00507 char  writeCol(EmbPattern *pattern, FILE* file);
00508 char  readCsd(EmbPattern *pattern, FILE* file);
00509 char  writeCsd(EmbPattern *pattern, FILE* file);
00510 char  readCsv(EmbPattern *pattern, FILE* file);
00511 char  writeCsv(EmbPattern *pattern, FILE* file);
00512 char  readDat(EmbPattern *pattern, FILE* file);
00513 char  writeDat(EmbPattern *pattern, FILE* file);
00514 char  readDem(EmbPattern *pattern, FILE* file);
00515 char  writeDem(EmbPattern *pattern, FILE* file);
00516 char  readDsb(EmbPattern *pattern, FILE* file);
00517 char  writeDsb(EmbPattern *pattern, FILE* file);
00518 char  readDst(EmbPattern *pattern, FILE* file);
00519 char  writeDst(EmbPattern *pattern, FILE* file);
00520 char  readDsz(EmbPattern *pattern, FILE* file);
00521 char  writeDsz(EmbPattern *pattern, FILE* file);
00522 char  readDxf(EmbPattern *pattern, FILE* file);
00523 char  writeDxf(EmbPattern *pattern, FILE* file);
00524 char  readEdr(EmbPattern *pattern, FILE* file);
00525 char  writeEdr(EmbPattern *pattern, FILE* file);
00526 char  readEmd(EmbPattern *pattern, FILE* file);
00527 char  writeEmd(EmbPattern *pattern, FILE* file);
00528 char  readExp(EmbPattern *pattern, FILE* file);
00529 char  writeExp(EmbPattern *pattern, FILE* file);
00530 char  readExy(EmbPattern *pattern, FILE* file);
00531 char  writeExy(EmbPattern *pattern, FILE* file);
00532 char  readEys(EmbPattern *pattern, FILE* file);
00533 char  writeEys(EmbPattern *pattern, FILE* file);
00534 char  readFxy(EmbPattern *pattern, FILE* file);
00535 char  writeFxy(EmbPattern *pattern, FILE* file);
00536 char  readGc(EmbPattern *pattern, FILE* file);
00537 char  writeGc(EmbPattern *pattern, FILE* file);
00538 char  readGnc(EmbPattern *pattern, FILE* file);
00539 char  writeGnc(EmbPattern *pattern, FILE* file);
00540 char  readGt(EmbPattern *pattern, FILE* file);
00541 char  writeGt(EmbPattern *pattern, FILE* file);
00542 char  readHus(EmbPattern *pattern, FILE* file);
00543 char  writeHus(EmbPattern *pattern, FILE* file);
00544 char  readInb(EmbPattern *pattern, FILE* file);
00545 char  writeInb(EmbPattern *pattern, FILE* file);
00546 char  readInf(EmbPattern *pattern, FILE* file);
00547 char  writeInf(EmbPattern *pattern, FILE* file);
00548 char  readJef(EmbPattern *pattern, FILE* file);
00549 char  writeJef(EmbPattern *pattern, FILE* file);
00550 char  readKsm(EmbPattern *pattern, FILE* file);
00551 char  writeKsm(EmbPattern *pattern, FILE* file);
00552 char  readMax(EmbPattern *pattern, FILE* file);
00553 char  writeMax(EmbPattern *pattern, FILE* file);
00554 char  readMit(EmbPattern *pattern, FILE* file);
00555 char  writeMit(EmbPattern *pattern, FILE* file);
00556 char  readNew(EmbPattern *pattern, FILE* file);
00557 char  writeNew(EmbPattern *pattern, FILE* file);
00558 char  readOfm(EmbPattern *pattern, FILE* file);
00559 char  writeOfm(EmbPattern *pattern, FILE* file);
00560 char  readPcd(EmbPattern *pattern, const char *fileName, FILE* file);
00561 char  writePcd(EmbPattern *pattern, FILE* file);
00562 char  readPcm(EmbPattern *pattern, FILE* file);
00563 char  writePcm(EmbPattern *pattern, FILE* file);
00564 char  readPcq(EmbPattern *pattern, const char *fileName, FILE* file);
00565 char  writePcq(EmbPattern *pattern, FILE* file);

```

```

00566 char readPcs(EmbPattern *pattern, const char *fileName, FILE* file);
00567 char writePcs(EmbPattern *pattern, FILE* file);
00568 char readPec(EmbPattern *pattern, const char *fileName, FILE* file);
00569 char writePec(EmbPattern *pattern, const char *fileName, FILE* file);
00570 char readPel(EmbPattern *pattern, FILE *file);
00571 char writePel(EmbPattern *pattern, FILE *file);
00572 char readPem(EmbPattern *pattern, FILE *file);
00573 char writePem(EmbPattern *pattern, FILE *file);
00574 char readPes(EmbPattern *pattern, const char *fileName, FILE* file);
00575 char writePes(EmbPattern *pattern, const char *fileName, FILE* file);
00576 char readPhb(EmbPattern *pattern, FILE* file);
00577 char writePhb(EmbPattern *pattern, FILE *file);
00578 char readPhc(EmbPattern *pattern, FILE* file);
00579 char writePhc(EmbPattern *pattern, FILE *file);
00580 char readPlt(EmbPattern *pattern, FILE* file);
00581 char writePlt(EmbPattern *pattern, FILE* file);
00582 char readRgb(EmbPattern *pattern, FILE* file);
00583 char writeRgb(EmbPattern *pattern, FILE* file);
00584 char readSew(EmbPattern *pattern, FILE* file);
00585 char writeSew(EmbPattern *pattern, FILE* file);
00586 char readShv(EmbPattern *pattern, FILE* file);
00587 char writeShv(EmbPattern *pattern, FILE *file);
00588 char readSst(EmbPattern *pattern, FILE* file);
00589 char writeSst(EmbPattern *pattern, FILE *file);
00590 char readStx(EmbPattern *pattern, FILE* file);
00591 char writeStx(EmbPattern *pattern, FILE *file);
00592 char readSvg(EmbPattern *pattern, FILE* file);
00593 char writeSvg(EmbPattern *pattern, FILE* file);
00594 char readT01(EmbPattern *pattern, FILE* file);
00595 char writeT01(EmbPattern *pattern, FILE* file);
00596 char readT09(EmbPattern *pattern, FILE* file);
00597 char writeT09(EmbPattern *pattern, FILE* file);
00598 char readTap(EmbPattern *pattern, FILE* file);
00599 char writeTap(EmbPattern *pattern, FILE* file);
00600 char readThr(EmbPattern *pattern, FILE* file);
00601 char writeThr(EmbPattern *pattern, FILE* file);
00602 char readTxt(EmbPattern *pattern, FILE* file);
00603 char writeTxt(EmbPattern *pattern, FILE* file);
00604 char readU00(EmbPattern *pattern, FILE* file);
00605 char writeU00(EmbPattern *pattern, FILE *file);
00606 char readU01(EmbPattern *pattern, FILE* file);
00607 char writeU01(EmbPattern *pattern, FILE *file);
00608 char readVip(EmbPattern *pattern, FILE* file);
00609 char writeVip(EmbPattern *pattern, FILE* file);
00610 char readVp3(EmbPattern *pattern, FILE* file);
00611 char writeVp3(EmbPattern *pattern, FILE* file);
00612 char readXxx(EmbPattern *pattern, FILE* file);
00613 char writeXxx(EmbPattern *pattern, FILE* file);
00614 char readZsk(EmbPattern *pattern, FILE* file);
00615 char writeZsk(EmbPattern *pattern, FILE* file);
00616
00617 extern const char imageWithFrame[38][48];
00618
00619 #endif

```

4.9 src/encoding.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "embroidery_internal.h"

```

Functions

- void [write_24bit](#) (FILE *file, int)
- [EmbColor embColor_fromHexStr](#) (char *val)
- void [reverse_byte_order](#) (void *b, int bytes)
- int [decode_t01_record](#) (unsigned char b[3], int *x, int *y, int *flags)
- void [encode_t01_record](#) (unsigned char b[3], int x, int y, int flags)
- int [encode_tajima_ternary](#) (unsigned char b[3], int x, int y)

- void [decode_tajima_ternary](#) (unsigned char b[3], int *x, int *y)
- void [pfaffEncode](#) (FILE *file, int dx, int dy, int flags)
- [EmbReal pfaffDecode](#) (unsigned char a1, unsigned char a2, unsigned char a3)
- unsigned char [mitEncodeStitch](#) ([EmbReal](#) value)
- int [mitDecodeStitch](#) (unsigned char value)
- int [decodeNewStitch](#) (unsigned char value)
- void [emblnt_read](#) (FILE *f, char *label, void *b, int mode)
- void [emblnt_write](#) (FILE *f, char *label, void *b, int mode)

4.9.1 Function Documentation

4.9.1.1 decode_t01_record()

```
int decode_t01_record (
    unsigned char b[3],
    int * x,
    int * y,
    int * flags )
```

Definition at line [74](#) of file [encoding.c](#).

4.9.1.2 decode_tajima_ternary()

```
void decode_tajima_ternary (
    unsigned char b[3],
    int * x,
    int * y )
```

Definition at line [229](#) of file [encoding.c](#).

4.9.1.3 decodeNewStitch()

```
int decodeNewStitch (
    unsigned char value )
```

Definition at line [337](#) of file [encoding.c](#).

4.9.1.4 embColor_fromHexStr()

```
EmbColor embColor_fromHexStr (
    char * val )
```

Definition at line [32](#) of file [encoding.c](#).

4.9.1.5 `embInt_read()`

```
void embInt_read (
    FILE * f,
    char * label,
    void * b,
    int mode )
```

Definition at line 350 of file [encoding.c](#).

4.9.1.6 `embInt_write()`

```
void embInt_write (
    FILE * f,
    char * label,
    void * b,
    int mode )
```

Definition at line 381 of file [encoding.c](#).

4.9.1.7 `encode_t01_record()`

```
void encode_t01_record (
    unsigned char b[3],
    int x,
    int y,
    int flags )
```

Definition at line 99 of file [encoding.c](#).

4.9.1.8 `encode_tajima_ternary()`

```
int encode_tajima_ternary (
    unsigned char b[3],
    int x,
    int y )
```

Definition at line 118 of file [encoding.c](#).

4.9.1.9 `mitDecodeStitch()`

```
int mitDecodeStitch (
    unsigned char value )
```

Definition at line 330 of file [encoding.c](#).

4.9.1.10 **mitEncodeStitch()**

```
unsigned char mitEncodeStitch (
    EmbReal value )
```

Definition at line 323 of file [encoding.c](#).

4.9.1.11 **pfaffDecode()**

```
EmbReal pfaffDecode (
    unsigned char a1,
    unsigned char a2,
    unsigned char a3 )
```

Definition at line 315 of file [encoding.c](#).

4.9.1.12 **pfaffEncode()**

```
void pfaffEncode (
    FILE * file,
    int dx,
    int dy,
    int flags )
```

Definition at line 295 of file [encoding.c](#).

4.9.1.13 **reverse_byte_order()**

```
void reverse_byte_order (
    void * b,
    int bytes )
```

Definition at line 56 of file [encoding.c](#).

4.9.1.14 **write_24bit()**

```
void write_24bit (
    FILE * file,
    int x )
```

Definition at line 660 of file [main.c](#).

4.10 encoding.c

[Go to the documentation of this file.](#)

```

00001 /*
00002 * This file is part of libembroidery.
00003 *
00004 * Copyright 2018-2022 The Embroidermodder Team
00005 * Licensed under the terms of the zlib license.
00006 *
00007 * This file contains all the read and write functions for the
00008 * library.
00009 */
00010
00011 #include <stdio.h>
00012 #include <stdlib.h>
00013 #include <string.h>
00014 #include <math.h>
00015
00016 #include "embroidery_internal.h"
00017
00018 void write_24bit(FILE* file, int);
00019
00020 /* ENCODING
00021 ****
00022 * The functions in this file are grouped together to aid the developer's
00023 * understanding of the similarities between the file formats. This also helps
00024 * reduce errors between reimplementation of the same idea.
00025 *
00026 * For example: the Tajima ternary encoding of positions is used by at least 4
00027 * formats and the only part that changes is the flag encoding.
00028 */
00029
00030 /* Converts a 6 digit hex string (I.E. "00FF00")
00031 into an EmbColor and returns it. */
00032 EmbColor embColor_fromHexStr(char* val) {
00033     EmbColor color;
00034     char r[3];
00035     char g[3];
00036     char b[3];
00037
00038     r[0] = val[0];
00039     r[1] = val[1];
00040     r[2] = 0;
00041
00042     g[0] = val[2];
00043     g[1] = val[3];
00044     g[2] = 0;
00045
00046     b[0] = val[4];
00047     b[1] = val[5];
00048     b[2] = 0;
00049
00050     color.r = (unsigned char)strtol(r, 0, 16);
00051     color.g = (unsigned char)strtol(g, 0, 16);
00052     color.b = (unsigned char)strtol(b, 0, 16);
00053
00054     return color;
00055 }
00056 void reverse_byte_order(void *b, int bytes)
00057 {
00058     char swap;
00059     if (bytes == 2) {
00060         swap = *((char*)b+0);
00061         *((char*)b+0) = *((char*)b+1);
00062         *((char*)b+1) = swap;
00063     }
00064     else {
00065         swap = *((char*)b+0);
00066         *((char*)b+0) = *((char*)b+3);
00067         *((char*)b+3) = swap;
00068         swap = *((char*)b+1);
00069         *((char*)b+1) = *((char*)b+2);
00070         *((char*)b+2) = swap;
00071     }
00072 }
00073
00074 int decode_t01_record(unsigned char b[3], int *x, int *y, int *flags) {
00075     decode_tajima_ternary(b, x, y);
00076
00077     if (b[2] == 0xF3) {
00078         *flags = END;
00079     }
00080     else {
00081         switch (b[2] & 0xC3) {
00082             case 0x03:

```

```
00083         *flags = NORMAL;
00084         break;
00085     case 0x83:
00086         *flags = TRIM;
00087         break;
00088     case 0xC3:
00089         *flags = STOP;
00090         break;
00091     default:
00092         *flags = NORMAL;
00093         break;
00094     }
00095 }
00096 return 1;
00097 }
00098
00099 void encode_t01_record(unsigned char b[3], int x, int y, int flags) {
00100     if (!encode_tajima_ternary(b, x, y)) {
00101         return;
00102     }
00103
00104     b[2] |= (unsigned char)3;
00105     if (flags & END) {
00106         b[0] = 0;
00107         b[1] = 0;
00108         b[2] = 0xF3;
00109     }
00110     if (flags & (JUMP | TRIM)) {
00111         b[2] = (unsigned char)(b[2] | 0x83);
00112     }
00113     if (flags & STOP) {
00114         b[2] = (unsigned char)(b[2] | 0xC3);
00115     }
00116 }
00117
00118 int encode_tajima_ternary(unsigned char b[3], int x, int y)
00119 {
00120     b[0] = 0;
00121     b[1] = 0;
00122     b[2] = 0;
00123
00124     /* cannot encode values > +121 or < -121. */
00125     if (x > 121 || x < -121) {
00126         printf("ERROR: format-t01.c encode_record(), ");
00127         printf("x is not in valid range [-121,121] , x = %d\n", x);
00128         return 0;
00129     }
00130     if (y > 121 || y < -121) {
00131         printf("ERROR: format-t01.c encode_record(), ");
00132         printf("y is not in valid range [-121,121] , y = %d\n", y);
00133         return 0;
00134     }
00135
00136     if (x >= +41) {
00137         b[2] |= 0x04;
00138         x -= 81;
00139     }
00140     if (x <= -41) {
00141         b[2] |= 0x08;
00142         x += 81;
00143     }
00144     if (x >= +14) {
00145         b[1] |= 0x04;
00146         x -= 27;
00147     }
00148     if (x <= -14) {
00149         b[1] |= 0x08;
00150         x += 27;
00151     }
00152     if (x >= +5) {
00153         b[0] |= 0x04;
00154         x -= 9;
00155     }
00156     if (x <= -5) {
00157         b[0] |= 0x08;
00158         x += 9;
00159     }
00160     if (x >= +2) {
00161         b[1] |= 0x01;
00162         x -= 3;
00163     }
00164     if (x <= -2) {
00165         b[1] |= 0x02;
00166         x += 3;
00167     }
00168     if (x >= +1) {
00169         b[0] |= 0x01;
```

```

00170         x -= 1;
00171     }
00172     if (x <= -1) {
00173         b[0] |= 0x02;
00174         x += 1;
00175     }
00176     if (x != 0) {
00177         printf("ERROR: format-dst.c encode_record(), ");
00178         printf("x should be zero yet x = %d\n", x);
00179         return 0;
00180     }
00181     if (y >= +41) {
00182         b[2] |= 0x20;
00183         y -= 81;
00184     }
00185     if (y <= -41) {
00186         b[2] |= 0x10;
00187         y += 81;
00188     }
00189     if (y >= +14) {
00190         b[1] |= 0x20;
00191         y -= 27;
00192     }
00193     if (y <= -14) {
00194         b[1] |= 0x10;
00195         y += 27;
00196     }
00197     if (y >= +5) {
00198         b[0] |= 0x20;
00199         y -= 9;
00200     }
00201     if (y <= -5) {
00202         b[0] |= 0x10;
00203         y += 9;
00204     }
00205     if (y >= +2) {
00206         b[1] |= 0x80;
00207         y -= 3;
00208     }
00209     if (y <= -2) {
00210         b[1] |= 0x40;
00211         y += 3;
00212     }
00213     if (y >= +1) {
00214         b[0] |= 0x80;
00215         y -= 1;
00216     }
00217     if (y <= -1) {
00218         b[0] |= 0x40;
00219         y += 1;
00220     }
00221     if (y != 0) {
00222         printf("ERROR: format-dst.c encode_record(), ");
00223         printf("y should be zero yet y = %d\n", y);
00224         return 0;
00225     }
00226     return 1;
00227 }
00228
00229 void decode_tajima_ternary(unsigned char b[3], int *x, int *y)
00230 {
00231     *x = 0;
00232     *y = 0;
00233     if (b[0] & 0x01) {
00234         *x += 1;
00235     }
00236     if (b[0] & 0x02) {
00237         *x -= 1;
00238     }
00239     if (b[0] & 0x04) {
00240         *x += 9;
00241     }
00242     if (b[0] & 0x08) {
00243         *x -= 9;
00244     }
00245     if (b[0] & 0x80) {
00246         *y += 1;
00247     }
00248     if (b[0] & 0x40) {
00249         *y -= 1;
00250     }
00251     if (b[0] & 0x20) {
00252         *y += 9;
00253     }
00254     if (b[0] & 0x10) {
00255         *y -= 9;
00256     }

```

```

00257     if (b[1] & 0x01) {
00258         *x += 3;
00259     }
00260     if (b[1] & 0x02) {
00261         *x -= 3;
00262     }
00263     if (b[1] & 0x04) {
00264         *x += 27;
00265     }
00266     if (b[1] & 0x08) {
00267         *x -= 27;
00268     }
00269     if (b[1] & 0x80) {
00270         *y += 3;
00271     }
00272     if (b[1] & 0x40) {
00273         *y -= 3;
00274     }
00275     if (b[1] & 0x20) {
00276         *y += 27;
00277     }
00278     if (b[1] & 0x10) {
00279         *y -= 27;
00280     }
00281     if (b[2] & 0x04) {
00282         *x += 81;
00283     }
00284     if (b[2] & 0x08) {
00285         *x -= 81;
00286     }
00287     if (b[2] & 0x20) {
00288         *y += 81;
00289     }
00290     if (b[2] & 0x10) {
00291         *y -= 81;
00292     }
00293 }
00294
00295 void pfaffEncode(FILE* file, int dx, int dy, int flags)
00296 {
00297     unsigned char flagsToWrite = 0;
00298
00299     if (!file) { printf("ERROR: format-pcs.c pcsEncode(), file argument is null\n"); return; }
00300
00301     write_24bit(file, dx);
00302     write_24bit(file, dy);
00303
00304     if (flags & STOP)
00305     {
00306         flagsToWrite |= 0x01;
00307     }
00308     if (flags & TRIM)
00309     {
00310         flagsToWrite |= 0x04;
00311     }
00312     fwrite(&flagsToWrite, 1, 1, file);
00313 }
00314
00315 EmbReal pfaffDecode(unsigned char a1, unsigned char a2, unsigned char a3) {
00316     int res = a1 + (a2 < 8) + (a3 < 16);
00317     if (res > 0xFFFF) {
00318         return -((~(res) & 0xFFFF) - 1);
00319     }
00320     return res;
00321 }
00322
00323 unsigned char mitEncodeStitch(EmbReal value) {
00324     if (value < 0) {
00325         return 0x80 | (unsigned char)(-value);
00326     }
00327     return (unsigned char)value;
00328 }
00329
00330 int mitDecodeStitch(unsigned char value) {
00331     if (value & 0x80) {
00332         return -(value & 0x1F);
00333     }
00334     return value;
00335 }
00336
00337 int decodeNewStitch(unsigned char value) {
00338     return (int)value;
00339 }
00340
00341 /* Read and write system for multiple byte types.
00342 *
00343 * The caller passes the function to read/write from, the

```

```

00344 * memory location as a void pointer and a mode identifier that describes
00345 * the type. This way we can abstract out the endianness of the
00346 * system running the library and don't have to maintain many functions,
00347 * just two.
00348 */
00349 void
00350 embInt_read(FILE* f, char *label, void *b, int mode)
00351 {
00352     int endian = mode & 0x01;
00353     int length = mode - endian;
00354     fread(b, 1, length, f);
00355     if (endian != ENDIAN_HOST) {
00356         reverse_byte_order(b, length);
00357     }
00358
00359     if (emb_verbose>1) {
00360         switch (mode) {
00361             case EMB_INT16_LITTLE:
00362                 printf("read int16_le %s: %d\n", label, *((short*)b));
00363                 break;
00364             case EMB_INT16_BIG:
00365                 printf("read int16_be %s: %d\n", label, *((short*)b));
00366                 break;
00367             case EMB_INT32_LITTLE:
00368                 printf("read int32_le %s: %d\n", label, *((int*)b));
00369                 break;
00370             case EMB_INT32_BIG:
00371                 printf("read int32_be %s: %d\n", label, *((int*)b));
00372                 break;
00373             default:
00374                 puts("ERROR: the mode supplied to fread_int is invalid.");
00375                 break;
00376         }
00377     }
00378 }
00379
00380 void
00381 embInt_write(FILE* f, char *label, void *b, int mode)
00382 {
00383     int endian = mode & 0x01;
00384     int length = mode - endian;
00385     if (endian != ENDIAN_HOST) {
00386         reverse_byte_order(b, length);
00387     }
00388     if (emb_verbose>1) {
00389         switch (mode) {
00390             case EMB_INT16_LITTLE:
00391                 printf("write int16_le %s: %d\n", label, *((short*)b));
00392                 break;
00393             case EMB_INT16_BIG:
00394                 printf("write int16_be %s: %d\n", label, *((short*)b));
00395                 break;
00396             case EMB_INT32_LITTLE:
00397                 printf("write int32_le %s: %d\n", label, *((int*)b));
00398                 break;
00399             case EMB_INT32_BIG:
00400                 printf("write int32_be %s: %d\n", label, *((int*)b));
00401                 break;
00402             default:
00403                 puts("ERROR: the mode supplied to fwrite_int is invalid.");
00404                 break;
00405         }
00406     }
00407     fwrite(b, 1, length, f);
00408 }
00409 }
00410

```

4.11 src/fill.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "embroidery_internal.h"

```

Functions

- int `lindenmayer_system` (`L_system` L, char *state, int iterations, int complete)
- void `embPattern_horizontal_fill` (`EmbPattern` *pattern, `EmblImage` *image, int threshold)
- void `embPattern_crossstitch` (`EmbPattern` *pattern, `EmblImage` *image, int threshold)
- int `hilbert_curve` (`EmbPattern` *pattern, int iterations)
- void `generate_dragon_curve` (char *state, int iterations)
- int `dragon_curve` (int iterations)
- void `embPolygon_reduceByDistance` (`EmbArray` *vertices, `EmbArray` *simplified, float distance)
- void `embPolygon_reduceByNth` (`EmbArray` *vertices, `EmbArray` *out, int nth)
- `EmbPattern` * `embPattern_combine` (`EmbPattern` *p1, `EmbPattern` *p2)
- void `embPattern_stitchArc` (`EmbPattern` *p, `EmbArc` arc, int thread_index, int style)
- void `embPattern_stitchCircle` (`EmbPattern` *p, `EmbCircle` circle, int thread_index, int style)
- void `embPattern_stitchEllipse` (`EmbPattern` *p, `EmbEllipse` ellipse, int thread_index, int style)
- void `embPattern_stitchPath` (`EmbPattern` *p, `EmbPath` rect, int thread_index, int style)
- void `embPattern_stitchPolygon` (`EmbPattern` *p, `EmbPolygon` rect, int thread_index, int style)
- void `embPattern_stitchPolyline` (`EmbPattern` *p, `EmbPolyline` rect, int thread_index, int style)
- void `embPattern_stitchRect` (`EmbPattern` *p, `EmbRect` rect, int thread_index, int style)
- void `embPattern_stitchText` (`EmbPattern` *p, `EmbRect` rect, int thread_index, int style)
- void `embPattern_convertGeometry` (`EmbPattern` *p)

Variables

- const char * `rules` [] = {"+BF-AFA-FB+", "-AF+BFB+FA-"}
 • `L_system hilbert_curve_l_system`

4.11.1 Function Documentation

4.11.1.1 dragon_curve()

```
int dragon_curve (
    int iterations )
```

Definition at line 345 of file `fill.c`.

4.11.1.2 embPattern_combine()

```
EmbPattern * embPattern_combine (
    EmbPattern * p1,
    EmbPattern * p2 )
```

Definition at line 854 of file `fill.c`.

4.11.1.3 embPattern_convertGeometry()

```
void embPattern_convertGeometry (
    EmbPattern * p )
```

Definition at line [987](#) of file [fill.c](#).

4.11.1.4 embPattern_crossstitch()

```
void embPattern_crossstitch (
    EmbPattern * pattern,
    EmbImage * image,
    int threshold )
```

Definition at line [239](#) of file [fill.c](#).

4.11.1.5 embPattern_horizontal_fill()

```
void embPattern_horizontal_fill (
    EmbPattern * pattern,
    EmbImage * image,
    int threshold )
```

Definition at line [212](#) of file [fill.c](#).

4.11.1.6 embPattern_stitchArc()

```
void embPattern_stitchArc (
    EmbPattern * p,
    EmbArc arc,
    int thread_index,
    int style )
```

Definition at line [875](#) of file [fill.c](#).

4.11.1.7 embPattern_stitchCircle()

```
void embPattern_stitchCircle (
    EmbPattern * p,
    EmbCircle circle,
    int thread_index,
    int style )
```

Definition at line [881](#) of file [fill.c](#).

4.11.1.8 embPattern_stitchEllipse()

```
void embPattern_stitchEllipse (
    EmbPattern * p,
    EmbEllipse ellipse,
    int thread_index,
    int style )
```

Definition at line 933 of file [fill.c](#).

4.11.1.9 embPattern_stitchPath()

```
void embPattern_stitchPath (
    EmbPattern * p,
    EmbPath rect,
    int thread_index,
    int style )
```

Definition at line 939 of file [fill.c](#).

4.11.1.10 embPattern_stitchPolygon()

```
void embPattern_stitchPolygon (
    EmbPattern * p,
    EmbPolygon rect,
    int thread_index,
    int style )
```

Definition at line 945 of file [fill.c](#).

4.11.1.11 embPattern_stitchPolyline()

```
void embPattern_stitchPolyline (
    EmbPattern * p,
    EmbPolyline rect,
    int thread_index,
    int style )
```

Definition at line 951 of file [fill.c](#).

4.11.1.12 embPattern_stitchRect()

```
void embPattern_stitchRect (
    EmbPattern * p,
    EmbRect rect,
    int thread_index,
    int style )
```

Definition at line [957](#) of file [fill.c](#).

4.11.1.13 embPattern_stitchText()

```
void embPattern_stitchText (
    EmbPattern * p,
    EmbRect rect,
    int thread_index,
    int style )
```

Definition at line [982](#) of file [fill.c](#).

4.11.1.14 embPolygon_reduceByDistance()

```
void embPolygon_reduceByDistance (
    EmbArray * vertices,
    EmbArray * simplified,
    float distance )
```

Definition at line [805](#) of file [fill.c](#).

4.11.1.15 embPolygon_reduceByNth()

```
void embPolygon_reduceByNth (
    EmbArray * vertices,
    EmbArray * out,
    int nth )
```

Definition at line [835](#) of file [fill.c](#).

4.11.1.16 generate_dragon_curve()

```
void generate_dragon_curve (
    char * state,
    int iterations )
```

Definition at line [324](#) of file [fill.c](#).

4.11.1.17 hilbert_curve()

```
int hilbert_curve (
    EmbPattern * pattern,
    int iterations )
```

Definition at line 268 of file [fill.c](#).

4.11.1.18 lindenmayer_system()

```
int lindenmayer_system (
    L_system L,
    char * state,
    int iterations,
    int complete )
```

Definition at line 27 of file [fill.c](#).

4.11.2 Variable Documentation

4.11.2.1 hilbert_curve_l_system

`L_system hilbert_curve_l_system`

Initial value:

```
= {
    'A', "AB", "F+-", (char**)rules
}
```

Definition at line 22 of file [fill.c](#).

4.11.2.2 rules

```
const char* rules[] = {"+BF-AFA-FB+", "-AF+BFB+FA-"};
```

Definition at line 20 of file [fill.c](#).

4.12 fill.c

[Go to the documentation of this file.](#)

```

00001 /*
00002 * This file is part of libembroidery.
00003 *
00004 * Copyright 2018-2022 The Embroidermodder Team
00005 * Licensed under the terms of the zlib license.
00006 *
00007 * This file contains all the read and write functions for the
00008 * library.
00009 *
00010 * FILL ALGORITHMS
00011 *****/
00012
00013 #include <stdio.h>
00014 #include <stdlib.h>
00015 #include <string.h>
00016 #include <math.h>
00017
00018 #include "embroidery_internal.h"
00019
00020 const char *rules[] = {"+BF-AFA-FB+", "-AF+BFB+FA-"};
00021
00022 L_system hilbert_curve_l_system = {
00023     'A', "AB", "F+-", (char**)rules
00024 };
00025
00026 /* This is a slow generation algorithm */
00027 int lindenmayer_system(L_system L, char *state, int iterations, int complete)
00028 {
00029     /* We know that the full length of the curve has to fit within
00030      * the number of stitches and we can cancel consecutive +-,-+
00031      * etc.
00032      *
00033
00034 Potential reference:
00035
00036 @book{Prusinkiewicz1996Mar,
00037     author = {Prusinkiewicz, Przemyslaw and Lindenmayer, Aristid and Hanan, J. S. and Fracchia, F. D.
00038     and Fowler, D. R. and de Boer, M. J. M. and Mercer, L.},
00039     title = {{The Algorithmic Beauty of Plants (The Virtual Laboratory)}},
00040     year = {1996},
00041     month = {Mar},
00042     publisher = {Springer}
00043 }
00044     */
00045     char *new_state;
00046     int j;
00047
00048     if (complete == 0) {
00049         state[0] = L.axiom;
00050         state[1] = 0;
00051         lindenmayer_system(L, state, iterations, complete+1);
00052         return 0;
00053     }
00054     new_state = state + MAX_STITCHES*5;
00055
00056     new_state[0] = 0;
00057
00058     /* replace letters using rules by copying to new_state */
00059     for (j=0; j < (int)strlen(state); j++) {
00060         if (state[j] >= 'A' && state[j] < 'F') {
00061             strcat(new_state, L.rules[state[j]-'A']);
00062         }
00063         if (state[j] == 'F') {
00064             strcat(new_state, "F");
00065         }
00066         if (state[j] == '+') {
00067             strcat(new_state, "+");
00068         }
00069         if (state[j] == '-') {
00070             strcat(new_state, "-");
00071         }
00072     }
00073     memcpy(state, new_state, strlen(new_state)+1);
00074
00075     if (complete < iterations) {
00076         lindenmayer_system(L, state, iterations, complete+1);
00077     }
00078     return 0;
00079 }
00080
00081 /* Remove points that lie in the middle of two short stitches that could

```

```

00082 * be one longer stitch. Repeat until none are found.
00083 */
00084 static void
00085 join_short_stitches(int *points, int *n_points, int width, int tolerance)
00086 {
00087     int found = 1;
00088     while (found > 0) {
00089         int i;
00090         found = 0;
00091         for (i=*n_points-2; i>=0; i--) {
00092             int st1 = points[i+1]%width - points[i]%width;
00093             int st2 = points[i+2]%width - points[i+1]%width;
00094             int same_line = (points[i+1]/width == points[i]/width)
00095                         && (points[i+2]/width == points[i+1]/width);
00096             if (st1 < tolerance && st2 < tolerance && same_line) {
00097                 found++;
00098                 break;
00099             }
00100         }
00101         if (found) {
00102             /* Remove the point. */
00103             i++;
00104             for (; i<*n_points; i++) {
00105                 points[i] = points[i+1];
00106             }
00107             (*n_points)--;
00108         }
00109     }
00110 }
00111
00112 /* Identify darker pixels to put stitches in.
00113 */
00114 static int *
00115 threshold_method(EmbImage *image, int *n_points,
00116                     int subsample_width, int subsample_height, int threshold)
00117 {
00118     int i, j;
00119     int *points;
00120     int height = image->height;
00121     int width = image->width;
00122     points = (int *)malloc((height/subsample_height)
00123                            *(width/subsample_width) * sizeof(int));
00124     *n_points = 0;
00125     for (i=0; i<height/subsample_height; i++)
00126         for (j=0; j<width/subsample_width; j++) {
00127             EmbColor color;
00128             int index = subsample_height*i*width+subsample_width*j;
00129             color.r = image->data[3*index+0];
00130             color.g = image->data[3*index+1];
00131             color.b = image->data[3*index+2];
00132             if (color.r+color.g+color.b < threshold) {
00133                 points[*n_points] = index;
00134                 (*n_points)++;
00135             }
00136         }
00137     return points;
00138 }
00139
00140 /* Greedy Algorithm
00141 * -----
00142 * For each point in the list find the shortest distance to
00143 * any possible neighbour, then perform a swap to make that
00144 * neighbour the next item in the list.
00145 *
00146 * To make the stitches lie more on one axis than the other
00147 * bias the distance operator to prefer horizontal direction.
00148 */
00149 static void
00150 greedy_algorithm(int *points, int n_points, int width, EmbReal bias)
00151 {
00152     int i, j;
00153     printf("points[0] = %d\n", points[0]);
00154     printf("n_points = %d\n", n_points);
00155     printf("width = %d\n", width);
00156     printf("bias = %f\n", bias);
00157
00158     for (i=0; i<n_points-1; i++) {
00159         int stor;
00160         EmbReal shortest = 1.0e20;
00161         int next = i+1;
00162         /* Find nearest neighbour. */
00163         int xl = points[i]%width;
00164         int yl = points[i]/width;
00165         for (j=i+1; j<n_points; j++) {
00166             int x, y;
00167             EmbReal distance;
00168             x = xl - (points[j]%width);

```

```

00169         if (x*x > shortest) {
00170             continue;
00171         }
00172         y = y1 - (points[j]/width);
00173         distance = x*x + bias*y*y;
00174         if (distance < shortest) {
00175             next = j;
00176             shortest = distance;
00177         }
00178     }
00179     if (i%100 == 0) {
00180         printf("%2.1f%\n", (100.0*i)/(1.0*n_points));
00181     }
00182     /* swap points */
00183     stor = points[next];
00184     points[next] = points[i+1];
00185     points[i+1] = stor;
00186 }
00187 }
00188
00189 static void
00190 save_points_to_pattern(
00191     EmbPattern *pattern, int *points, int n_points, EmbReal scale, int width, int height)
00192 {
00193     int i;
00194     for (i=0; i<n_points; i++) {
00195         int x, y;
00196         x = points[i]%width;
00197         y = height - points[i]/width;
00198         embPattern_addStitchAbs(pattern, scale*x, scale*y, NORMAL, 0);
00199     }
00200 }
00201
00202 /* Uses a threshold method to determine where to put
00203 * lines in the fill.
00204 *
00205 * Needs to pass a "donut test", i.e. an image with black pixels where:
00206 *   10 < x*x + y*y < 20
00207 * over the area (-30, 30) x (-30, 30).
00208 *
00209 * Use render then image difference to see how well it passes.
00210 */
00211 void
00212 embPattern_horizontal_fill(EmbPattern *pattern, EmbImage *image, int threshold)
00213 {
00214     /* Size of the crosses in millimeters. */
00215     EmbReal scale = 0.1;
00216     int sample_w = 3;
00217     int sample_h = 3;
00218     EmbReal bias = 1.2;
00219     int *points;
00220     int n_points;
00221
00222     points = threshold_method(image, &n_points, sample_w, sample_h, threshold);
00223     greedy_algorithm(points, n_points, image->width, bias);
00224     join_short_stitches(points, &n_points, image->width, 40);
00225     save_points_to_pattern(pattern, points, n_points, scale, image->width, image->height);
00226
00227     embPattern_end(pattern);
00228     free(points);
00229 }
00230
00231 /* Uses a threshold method to determine where to put
00232 * crosses in the fill.
00233 *
00234 * To improve this, we can remove the vertical stitches when two crosses
00235 * neighbour. Currently the simple way to do this is to chain crosses
00236 * that are neighbours exactly one ahead.
00237 */
00238 void
00239 embPattern_crossstitch(EmbPattern *pattern, EmbImage *image, int threshold)
00240 {
00241     int i;
00242     /* Size of the crosses in millimeters. */
00243     EmbReal scale = 0.1;
00244     int sample_w = 5;
00245     int sample_h = 5;
00246     EmbReal bias = 1.0;
00247     int *points;
00248     int n_points;
00249     int width = 1000;
00250     points = threshold_method(image, &n_points, sample_w, sample_h, threshold);
00251     greedy_algorithm(points, n_points, width, bias);
00252
00253     for (i=0; i<n_points; i++) {
00254         EmbReal x, y;
00255         x = points[i]%width;

```

```

00256     y = points[i]/width;
00257     printf("%f %f\n", x, y);
00258     embPattern_addStitchAbs(pattern, scale*x, scale*y, NORMAL, 0);
00259     embPattern_addStitchAbs(pattern, scale*(x+sample_w), scale*(y+sample_h), NORMAL, 0);
00260     embPattern_addStitchAbs(pattern, scale*x, scale*(y+sample_h), NORMAL, 0);
00261     embPattern_addStitchAbs(pattern, scale*(x+sample_w), scale*y, NORMAL, 0);
00262 }
00263
00264 embPattern_end(pattern);
00265 }
00266
00267 int
00268 hilbert_curve(EmbPattern *pattern, int iterations)
00269 {
00270     /*
00271     https://en.wikipedia.org/wiki/Hilbert_curve
00272
00273     Using the Lindenmayer System, so we can save work across
00274     different functions.
00275     */
00276     char *state;
00277     int i, position[2], direction;
00278     EmbReal scale = 1.0;
00279
00280     /* Make the n-th iteration. */
00281     state = malloc(MAX_STITCHES*10);
00282     lindenmayer_system(hilbert_curve_l_system, state, iterations, 0);
00283
00284     /* Convert to an embroidery pattern. */
00285     position[0] = 0;
00286     position[1] = 0;
00287     direction = 0;
00288
00289     for (i = 0; i < (int)strlen(state); i++) {
00290         if (state[i] == '+') {
00291             direction = (direction + 1) % 4;
00292             continue;
00293         }
00294         if (state[i] == '-') {
00295             direction = (direction + 3) % 4;
00296             continue;
00297         }
00298         if (state[i] == 'F') {
00299             int flags = NORMAL;
00300             switch (direction) {
00301                 case 0:
00302                     default:
00303                         position[0]--;
00304                         break;
00305                 case 1:
00306                     position[1]++;
00307                     break;
00308                 case 2:
00309                     position[0]++;
00310                     break;
00311                 case 3:
00312                     position[1]--;
00313                     break;
00314             }
00315             embPattern_addStitchAbs(pattern, position[0]*scale, position[1]*scale, flags, 0);
00316         }
00317     }
00318     free(state);
00319     embPattern_end(pattern);
00320     return 0;
00321 }
00322
00323 /* using the "paper folding" method (find citation) */
00324 void generate_dragon_curve(char *state, int iterations)
00325 {
00326     int i, length;
00327     if (iterations == 1) {
00328         state[0] = 'R';
00329         state[1] = 0;
00330         return;
00331     }
00332     length = strlen(state);
00333     for (i=length-1; i>=0; i--) {
00334         state[2*i+1] = state[i];
00335         if (i%2 == 0) {
00336             state[2*i] = 'R';
00337         } else {
00338             state[2*i] = 'L';
00339         }
00340     }
00341     state[2*length+1] = 0;
00342     generate_dragon_curve(state, iterations-1);

```

```

00343 }
00344
00345 int dragon_curve(int iterations)
00346 {
00347     char *state;
00348     if (iterations > 10) {
00349         puts("The dragon curve is only supported up to 10 iterations.");
00350         return 0;
00351     }
00352     state = malloc(1<<(iterations+1));
00353     generate_dragon_curve(state, iterations);
00354     free(state);
00355     return 1;
00356 }
00357
00358 #if 0
00359 StitchBlock* BreakIntoColorBlocks(EmbPattern *pattern)
00360 {
00361     int i;
00362     int sa2 = new StitchBlock();
00363     int oldColor = pattern->stitch_list->stitch[0].color;
00364     int color = pattern.ColorList[oldColor];
00365     sa2.Thread = new Thread(color.Red, color.Blue, color.Green);
00366     for (i = 0; i < pattern->stitch_list->count; i++) {
00367         EmbStitch s = pattern->stitch_list->stitch[i];
00368         if (s.color != oldColor) {
00369             yield return sa2;
00370             sa2 = new StitchBlock();
00371             color = pattern.ColorList[s.ColorIndex];
00372             sa2.Thread = new Thread(color.Red, color.Blue, color.Green);
00373             oldColor = s.ColorIndex;
00374         }
00375         int vs = new VectorStitch { Xy = new Point(s.X, s.Y), Color = s.ColorIndex };
00376         sa2.Stitches.Add(vs);
00377     }
00378     yield return sa2;
00379 }
00380
00381
00382
00383 StitchBlock * BreakIntoSeparateObjects(EmbStitchBlock* blocks)
00384 {
00385     int i, block;
00386     EmbReal previousAngle = 0.0;
00387     for (block=0; block<blocks->length; block++) {
00388         int stitches = new List<VectorStitch>();
00389         block.Stitches[0].Type = VectorStitchType.Contour;
00390         block.Stitches[block.Stitches.Count - 1].Type = VectorStitchType.Contour;
00391
00392         for (int i = 0; i < block.Stitches.Count - 2; i++) { /* step 0 */
00393             EmbReal dx = (embVector_relativeX(block.Stitches[i].Xy, block.Stitches[i + 1].Xy,
00394                         block.Stitches[i + 2].Xy));
00395             block.Stitches[i + 1].Type = dx <= 0 ? VectorStitchType.Run : VectorStitchType.Contour;
00396             block.Stitches[i].Angle = GetAngle(block.Stitches[i], block.Stitches[i + 1]);
00397             stitches.Add(block.Stitches[i].Clone());
00398             if (i > 0) {
00399                 if ((block.Stitches[i].Type == VectorStitchType.Contour) &&
00400                     fabs(block.Stitches[i].Angle - previousAngle) > (20/180*embConstantPi)) {
00401                     yield return
00402                         new StitchBlock
00403                         {
00404                             Stitches = stitches,
00405                             Angle = stitches.Average(x => x.Angle),
00406                             Thread = new Thread(block.Thread.Red, block.Thread.Blue,
00407                                     block.Thread.Green);
00408                         };
00409                         stitches = new List<VectorStitch>();
00410                     }
00411
00412             /* step 1 */
00413             for (i = 1; i < sa.Stitches.Count - 3; i++) {
00414                 if (sa.Stitches[i + 1].Type == VectorStitchType.Contour) {
00415                     float dy = embVector_relativeY(sa[i + 1].XY, sa[i + 2].XY, sa[i + 3].XY);
00416                     float dy2 = embVector_relativeY(sa[i].XY, sa[i + 1].XY, sa[i + 2].XY);
00417                     float dy3 = embVector_relativeY(sa[i + 2].XY, sa[i + 3].XY, sa[i + 4].XY);
00418                     if (dy)
00419                         if (sa.Stitches[i - 1].Type == VectorStitchType.Run || sa.Stitches[i + 1].Type ==
00420                             VectorStitchType.Run) {
00421                             sa.Stitches[i].Type = VectorStitchType.Tatami;
00422                         }
00423                     else {
00424                         sa.Stitches[i].Type = VectorStitchType.Satin;
00425                     }
00426                 }
00427             }
00428         }
00429     }
00430 }
```

```

00426         }
00427     }
00428 }
00429
00430 StitchObject * FindOutline(EmbStitchBlock* stitchData)
00431 {
00432     int currColorIndex = 0, sa;
00433     int pOdd = new List<Point>();
00434     int pEven = new List<Point>();
00435     for (sa=0; sa<stitchData->count; sa++) {
00436         if (sa.Stitches.Count > 0) {
00437             sa.Stitches[0].Type = VectorStitchType.Contour;
00438             sa.Stitches[sa.Stitches.Count - 1].Type = VectorStitchType.Contour;
00439             /* step 0 */
00440             for (int i = 0; i < sa.Stitches.Count - 2; i++) {
00441                 float dx = (GetRelativeX(sa.Stitches[i].Xy, sa.Stitches[i + 2].Xy));
00442                 sa.Stitches[i + 1].Type = dx <= 0 ? VectorStitchType.Run : VectorStitchType.Contour;
00443                 sa.Stitches[i].Angle = embVector_angle(sa.Stitches[i], sa.Stitches[i + 1]);
00444             }
00445             /* step 1 */
00446             for (int i = 1; i < sa.Stitches.Count - 3; i++) {
00447                 if (sa.Stitches[i + 1].Type == VectorStitchType.Contour) {
00448                     float dy = embVector_relativeY(sa[i + 1].XY, sa[i + 2].XY, sa[i + 3].XY);
00449                     float dy2 = embVector_relativeY(sa[i].XY, sa[i + 1].XY, sa[i + 2].XY);
00450                     float dy3 = embVector_relativeY(sa[i + 2].XY, sa[i + 3].XY, sa[i + 4].XY);
00451                     if (dy)
00452                         if (sa.Stitches[i - 1].Type == VectorStitchType.Run || sa.Stitches[i + 1].Type ==
00453                             VectorStitchType.Run) {
00454                             sa.Stitches[i].Type = VectorStitchType.Tatami;
00455                         }
00456                         else {
00457                             sa.Stitches[i].Type = VectorStitchType.Satin;
00458                         }
00459                     }
00460                 }
00461             }
00462
00463             int oddEven = 0;
00464             foreach (VectorStitch t in sa.Stitches) {
00465                 if ((t.Type == VectorStitchType.Contour) && (oddEven % 2) == 0) {
00466                     pEven.Add(t.Xy);
00467
00468                     oddEven++;
00469                 }
00470                 else if ((t.Type == VectorStitchType.Contour) && (oddEven % 2) == 1) {
00471                     pOdd.Add(t.Xy);
00472                     oddEven++;
00473                 }
00474             }
00475             currColorIndex++;
00476             int so = new StitchObject { SideOne = pEven, SideTwo = pOdd, ColorIndex = currColorIndex };
00477             yield return so;
00478             pEven = new List<Point>();
00479             pOdd = new List<Point>();
00480             /* break; */
00481         }
00482     }
00483
00484 EmbPattern DrawGraphics(EmbPattern p) {
00485     int stitchData = BreakIntoColorBlocks(p);
00486
00487     int outBlock = new List<StitchBlock>(BreakIntoSeparateObjects(stitchData));
00488     foreach(var block in stitchData) {
00489         foreach (var stitch in block.Stitches) {
00490             if (stitch.Angle != 0) {
00491                 int aaa = 1;
00492             }
00493         }
00494     }
00495     int xxxxx = outBlock;
00496     int objectsFound = FindOutline(stitchData);
00497     int outPattern = new Pattern();
00498     outPattern.AddColor(new Thread(255, 0, 0, "none", "None"));
00499     int colorIndex = outPattern.ColorList.Count - 1;
00500     int r = new Random();
00501     foreach (StitchObject stitchObject in objectsFound) {
00502         if (stitchObject.SideOne.Count > 1 && stitchObject.SideTwo.Count > 1) {
00503             outPattern.AddColor(new Thread((byte) (r.Next()%256), (byte) (r.Next()%256), (byte)
00504             (r.Next()%256),
00505                         "none", "None"));
00506             colorIndex++;
00507             outPattern.AddStitchRelative(0, 0, StitchTypes.Stop);
00508             int points = stitchObject.Generate2(75);
00509             foreach (var point in points) {
00510                 outPattern.AddStitchAbsolute(point.X, point.Y, StitchTypes.Normal);
00511             }
00512         }
00513     }
00514 }

```

```
00510         }
00511         break;
00512         StitchObject stitchObject = objectsFound[1]);
00513         if (stitchObject.SideOne.Count > 0) {
00514             outPattern.stitch_list.Add(new Stitch(stitchObject.SideOne[0].X,
00515                                         stitchObject.SideOne[0].Y,
00516                                         StitchType.Jump, colorIndex));
00517         }
00518         foreach (Point t in stitchObject.SideOne) {
00519             outPattern.stitch_list.Add(new Stitch(t.X, t.Y,
00520                                         StitchType.Normal, colorIndex));
00521         }
00522         foreach (Point t in stitchObject.SideTwo) {
00523             outPattern.stitch_list.Add(new Stitch(t.X, t.Y,
00524                                         StitchType.Normal, colorIndex));
00525         }
00526         break;
00527     }
00528 }
00529 outPattern.AddStitchRelative(0, 0, StitchTypes.End);
00530 return outPattern;
00531 /*
00532 return (SimplifyOutline(outPattern));
00533 */
00534 }
00535
00536 EmbPattern SimplifyOutline(EmbPattern pattern)
00537 {
00538     int v = new Vertices();
00539     v.AddRange(pattern.stitch_list.Select(point => new Vector2(point.X, point.Y)));
00540     int output = SimplifyTools.DouglasPeuckerSimplify(v, 10);
00541     int patternOut = new Pattern();
00542     foreach (var color in pattern.ColorList)
00543     {
00544         patternOut.AddColor(color);
00545     }
00546
00547     foreach (var vertex in output)
00548     {
00549         patternOut.AddStitchAbsolute(vertex.X, vertex.Y, StitchTypes.Normal);
00550     }
00551     patternOut.AddStitchRelative(0, 0, StitchTypes.End);
00552     return patternOut;
00553 }
00554
00555 bool[] _usePt;
00556 EmbReal _distanceTolerance;
00557
00558 /* Removes all collinear points on the polygon. */
00559 Vertices CollinearSimplify(Vertices vertices, float collinearityTolerance)
00560 {
00561     /* We can't simplify polygons under 3 vertices */
00562     if (vertices.Count < 3)
00563         return vertices;
00564
00565     int simplified = new Vertices();
00566
00567     for (int i = 0; i < vertices.Count; i++) {
00568         int prevId = vertices.PreviousIndex(i);
00569         int nextId = vertices.NextIndex(i);
00570
00571         Vector2 prev = vertices[prevId];
00572         Vector2 current = vertices[i];
00573         Vector2 next = vertices[nextId];
00574
00575         /* If they collinear, continue */
00576         if (embVector_collinear(ref prev, ref current, ref next, collinearityTolerance))
00577             continue;
00578
00579         simplified.Add(current);
00580     }
00581
00582     return simplified;
00583 }
00584
00585
00586 /* Removes all collinear points on the polygon.
00587 * Has a default bias of 0
00588 *
00589 * param vertices: The polygon that needs simplification.
00590 * returns: A simplified polygon.
00591 */
00592 Vertices CollinearSimplify(Vertices vertices)
00593 {
00594     return CollinearSimplify(vertices, 0);
00595 }
```

```

00596 /*
00597  * Ramer-Douglas-Peucker polygon simplification algorithm.
00598  * This is the general recursive version that does not use the
00599  * speed-up technique by using the Melkman convex hull.
00600  * If you pass in 0, it will remove all collinear points.
00601 */
00602 Vertices DouglasPeuckerSimplify(Vertices vertices, float distanceTolerance)
00603 {
00604     _distanceTolerance = distanceTolerance;
00605
00606     _usePt = new bool[vertices.Count];
00607     for (int i = 0; i < vertices.Count; i++)
00608     {
00609         _usePt[i] = true;
00610     }
00611
00612     SimplifySection(vertices, 0, vertices.Count - 1);
00613     int result = new Vertices();
00614     result.AddRange(vertices.Where((t, i) => _usePt[i]));
00615     return result;
00616 }
00617
00618 void SimplifySection(Vertices vertices, int i, int j)
00619 {
00620     if ((i + 1) == j)
00621         return;
00622
00623     Vector2 a = vertices[i];
00624     Vector2 b = vertices[j];
00625     EmbReal maxDistance = -1.0;
00626     int maxIndex = i;
00627     for (int k = i + 1; k < j; k++)
00628     {
00629         EmbReal distance = DistancePointLine(vertices[k], a, b);
00630
00631         if (distance > maxDistance)
00632         {
00633             maxDistance = distance;
00634             maxIndex = k;
00635         }
00636     }
00637     if (maxDistance <= _distanceTolerance) {
00638         for (int k = i + 1; k < j; k++) {
00639             _usePt[k] = 0;
00640         }
00641     }
00642     else {
00643         SimplifySection(vertices, i, maxIndex);
00644         SimplifySection(vertices, maxIndex, j);
00645     }
00646 }
00647
00648 EmbReal DistancePointLine(EmbVector p, EmbVector a, EmbVector b)
00649 {
00650     /* if start == end, then use point-to-point distance */
00651     if (a.X == b.X && a.Y == b.Y)
00652         return DistancePointPoint(p, a);
00653
00654     /* otherwise use comp.graphics.algorithms Frequently Asked Questions method */
00655     /* (1)          AC dot AB
00656      r =  -----
00657      ||AB||^2
00658
00659      r has the following meaning:
00660      r=0 Point = A
00661      r=1 Point = B
00662      r<0 Point is on the backward extension of AB
00663      r>1 Point is on the forward extension of AB
00664      0<r<1 Point is interior to AB
00665 */
00666
00667     EmbReal r = ((p.X - a.X) * (b.X - a.X) + (p.Y - a.Y) * (b.Y - a.Y))
00668     /
00669     ((b.X - a.X) * (b.X - a.X) + (b.Y - a.Y) * (b.Y - a.Y));
00670
00671     if (r <= 0.0) return DistancePointPoint(p, a);
00672     if (r >= 1.0) return DistancePointPoint(p, b);
00673
00674
00675     /* (2)
00676      (Ay-Cy) (Bx-Ax)-(Ax-Cx) (By-Ay)
00677      s = -----
00678      Curve^2
00679
00680      Then the distance from C to Point = |s|*Curve.
00681 */
00682

```

```

00683     EmbReal s = ((a.Y - p.Y) * (b.X - a.X) - (a.X - p.X) * (b.Y - a.Y))
00684             /
00685             ((b.X - a.X) * (b.X - a.X) + (b.Y - a.Y) * (b.Y - a.Y));
00686
00687     return fabs(s) * sqrt((b.X - a.X) * (b.X - a.X) + (b.Y - a.Y) * (b.Y - a.Y));
00688 }
00689
00690 /* From physics2d.net */
00691 public Vertices ReduceByArea(Vertices vertices, float areaTolerance)
00692 {
00693     if (vertices.Count <= 3)
00694         return vertices;
00695
00696     if (areaTolerance < 0)
00697     {
00698         throw new ArgumentOutOfRangeException("areaTolerance", "must be equal to or greater than
zero.");
00699     }
00700
00701     int result = new Vertices();
00702     Vector2 v3;
00703     Vector2 v1 = vertices[vertices.Count - 2];
00704     Vector2 v2 = vertices[vertices.Count - 1];
00705     areaTolerance *= 2;
00706     for (int index = 0; index < vertices.Count; ++index, v2 = v3)
00707     {
00708         if (index == vertices.Count - 1)
00709         {
00710             if (result.Count == 0)
00711             {
00712                 throw new ArgumentOutOfRangeException("areaTolerance", "The tolerance is too high!");
00713             }
00714             v3 = result[0];
00715         }
00716         else
00717         {
00718             v3 = vertices[index];
00719         }
00720         float old1, old2, new1;
00721         MathUtils.Cross(ref v1, ref v2, out old1);
00722         MathUtils.Cross(ref v2, ref v3, out old2);
00723         MathUtils.Cross(ref v1, ref v3, out new1);
00724         if (fabs(new1 - (old1 + old2)) > areaTolerance)
00725         {
00726             result.Add(v2);
00727             v1 = v2;
00728         }
00729     }
00730     return result;
00731 }
00732
00733 /* From Eric Jordan's convex decomposition library.
00734 * Merges all parallel edges in the list of vertices.
00735 */
00736 void MergeParallelEdges(EmbArray *vertices, float tolerance)
00737 {
00738     int i;
00739     if (vertices.Count <= 3) {
00740         /* Can't do anything useful here to a triangle. */
00741         return;
00742     }
00743
00744     int mergeMe = new bool[vertices.Count];
00745     int newNVertices = vertices.Count;
00746
00747     /* Gather points to process */
00748     for (i = 0; i < vertices->count; i++) {
00749         EmbVector delta0, delta1;
00750         int lower = (i == 0) ? (vertices.Count - 1) : (i - 1);
00751         int upper = (i == vertices.Count - 1) ? (0) : (i + 1);
00752
00753         delta0 = embVector_subtract(vertices[i], vertices[lower]);
00754         delta1 = embVector_subtract(vertices[upper], vertices[i]);
00755         float norm0 = embVector_length(delta0);
00756         float norm1 = embVector_length(delta1);
00757
00758         if (!(norm0 > 0.0f && norm1 > 0.0f) && newNVertices > 3) {
00759             /* Merge identical points */
00760             mergeMe[i] = 1;
00761             newNVertices--;
00762         }
00763
00764         embVector_normalize(delta0, &delta0);
00765         embVector_normalize(delta1, &delta1);
00766         float cross = embVector_cross(delta0, delta1);
00767         float dot = embVector_dot(delta0, delta1);
00768

```

```

00769     if (fabs(cross) < tolerance && dot > 0 && newNVertices > 3) {
00770         mergeMe[i] = 1;
00771         newNVertices--;
00772     }
00773     else {
00774         mergeMe[i] = 0;
00775     }
00776 }
00777
00778 if (newNVertices == vertices.Count || newNVertices == 0)
00779     return;
00780
00781 int currIndex = 0;
00782
00783 /* Copy the vertices to a new list and clear the old */
00784 int oldVertices = new Vertices(vertices);
00785 vertices.Clear();
00786
00787 for (i = 0; i < oldVertices.Count; i++) {
00788     if (mergeMe[i] || newNVertices == 0 || currIndex == newNVertices)
00789         continue;
00790
00791     vertices.Add(oldVertices[i]);
00792     currIndex++;
00793 }
00794 }
00795 #endif
00796
00797 void embPolygon_reduceByDistance(EmbArray *vertices, EmbArray *simplified, float distance);
00798 void embPolygon_reduceByNth(EmbArray *vertices, EmbArray *out, int nth);
00799
00800 /* Reduces the polygon by distance.
00801 */
00802 /* This is a non-destructive function, so the caller is responsible for
00803 * freeing "vertices" if they choose to keep "simplified".
00804 */
00805 void embPolygon_reduceByDistance(EmbArray *vertices, EmbArray *simplified, float distance)
00806 {
00807     int i;
00808     /* We can't simplify polygons under 3 vertices */
00809     if (vertices->count < 3) {
00810         embArray_copy(simplified, vertices);
00811         return;
00812     }
00813
00814     for (i = 0; i < vertices->count; i++) {
00815         EmbVector delta;
00816         int nextId = (i + 1) % vertices->count;
00817
00818         delta = embVector_subtract(
00819             vertices->geometry[nextId].object.vector,
00820             vertices->geometry[i].object.vector);
00821
00822         /* If they are closer than the distance, continue */
00823         if (embVector_length(delta) < distance) {
00824             continue;
00825         }
00826
00827         embArray_addVector(simplified, vertices->geometry[i].object.vector);
00828     }
00829 }
00830
00831 /* Reduces the polygon by removing the Nth vertex in the vertices list.
00832 * This is a non-destructive function, so the caller is responsible for
00833 * freeing vertices if they choose to keep out.
00834 */
00835 void embPolygon_reduceByNth(EmbArray *vertices, EmbArray *out, int nth)
00836 {
00837     int i;
00838     /* We can't simplify polygons under 3 vertices */
00839     if (vertices->count < 3) {
00840         embArray_copy(out, vertices);
00841         return;
00842     }
00843
00844     for (i=0; i<vertices->count; i++) {
00845         if (i!=nth) {
00846             embArray_addVector(out, vertices->geometry[i].object.vector);
00847         }
00848     }
00849 }
00850
00851 /*
00852 */
00853 EmbPattern *
00854 embPattern_combine(EmbPattern *p1, EmbPattern *p2)
00855 {

```

```

00856     int i;
00857     EmbPattern *out = embPattern_create();
00858     for (i=0; i<p1->stitch_list->count; i++) {
00859         embArray_addStitch(out->stitch_list, p1->stitch_list->stitch[i]);
00860     }
00861     for (i=0; i<p2->stitch_list->count; i++) {
00862         embArray_addStitch(out->stitch_list, p2->stitch_list->stitch[i]);
00863     }
00864     /* These need to be merged, not appended. */
00865     for (i=0; i<p1->thread_list->count; i++) {
00866         embPattern_addThread(out, p1->thread_list->thread[i]);
00867     }
00868     for (i=0; i<p2->thread_list->count; i++) {
00869         embPattern_addThread(out, p2->thread_list->thread[i]);
00870     }
00871     return out;
00872 }
00873
00874 void
00875 embPattern_stitchArc(EmbPattern *p, EmbArc arc, int thread_index, int style)
00876 {
00877
00878 }
00879
00880 void
00881 embPattern_stitchCircle(EmbPattern *p, EmbCircle circle, int thread_index, int style)
00882 {
00883     /* style determines:
00884      *   stitch density
00885      *   fill pattern
00886      *   outline or fill
00887      *
00888      * For now it's a straight fill of 1000 stitches of the whole object by
00889      * default.
00890      *
00891      * Consider the intersection of a line in direction "d" that passes through
00892      * the disc with center "c", radius "r". The start and end points are:
00893      *
00894      *   $(c-r(d/|d|), c + r(d/|d|))$
00895      *
00896      * Lines that are above and below this with an even separation $$ can be
00897      * found by taking the point on the line to be $sn$ where the $n$ is the
00898      * unit normal vector to $d$ and the vector to be $d$ again. The
00899      * intersection points are therefore a right angled triangle, with one side
00900      * $r$, another $s$ and the third the length to be solved, by Pythagoras we
00901      * have:
00902      *
00903      *   $(c + sn - \sqrt{r^2-s^2}(d/|d|), c + sn + \sqrt{r^2-s^2}(d/|d|))$
00904      *
00905      * repeating this process gives us all the end points and the fill only
00906      * alters these lines by splitting the ones longer than some tolerance.
00907      */
00908     float s;
00909     float separation = 0.1;
00910     EmbVector direction = {1.0, 1.0};
00911     EmbVector normal = {-1.0, 1.0};
00912     embVector_normalize(direction, &direction);
00913     embVector_normalize(normal, &normal);
00914     for (s=-circle.radius; s<circle.radius; s += separation) {
00915         EmbLine line;
00916         float length = sqrt(circle.radius*circle.radius - s*s);
00917         EmbVector scaled;
00918         embVector_multiply(normal, s, &scaled);
00919         line.start = embVector_add(circle.center, scaled);
00920         embVector_multiply(direction, length, &scaled);
00921         line.start = embVector_subtract(line.start, scaled);
00922         embVector_multiply(normal, s, &scaled);
00923         line.end = embVector_add(circle.center, scaled);
00924         embVector_multiply(direction, length, &scaled);
00925         line.end = embVector_add(line.end, scaled);
00926         /* Split long stitches here. */
00927         embPattern_addStitchAbs(p, line.start.x, line.start.y, NORMAL, thread_index);
00928         embPattern_addStitchAbs(p, line.end.x, line.end.y, NORMAL, thread_index);
00929     }
00930 }
00931
00932 void
00933 embPattern_stitchEllipse(EmbPattern *p, EmbEllipse ellipse, int thread_index, int style)
00934 {
00935
00936 }
00937
00938 void
00939 embPattern_stitchPath(EmbPattern *p, EmbPath rect, int thread_index, int style)
00940 {
00941
00942 }

```

```

00943
00944 void
00945 embPattern_stitchPolygon(EmbPattern *p, EmbPolygon rect, int thread_index, int style)
00946 {
00947
00948 }
00949
00950 void
00951 embPattern_stitchPolyline(EmbPattern *p, EmbPolyline rect, int thread_index, int style)
00952 {
00953
00954 }
00955
00956 void
00957 embPattern_stitchRect(EmbPattern *p, EmbRect rect, int thread_index, int style)
00958 {
00959     /* Here we just stitch the rectangle in the direction of it's longer side. */
00960     EmbReal separation = 0.1;
00961     EmbReal width = rect.right - rect.left;
00962     EmbReal height = rect.bottom - rect.top;
00963     if (width > height) {
00964         float s;
00965         for (s=rect.top; s<rect.bottom; s += seperation) {
00966             /* Split long stitches here. */
00967             embPattern_addStitchAbs(p, rect.top, s, NORMAL, thread_index);
00968             embPattern_addStitchAbs(p, rect.bottom, s, NORMAL, thread_index);
00969         }
00970     } else {
00971         float s;
00972         for (s=rect.left; s<rect.right; s += seperation) {
00973             /* Split long stitches here. */
00974             embPattern_addStitchAbs(p, s, rect.left, NORMAL, thread_index);
00975             embPattern_addStitchAbs(p, s, rect.right, NORMAL, thread_index);
00976         }
00977     }
00978 }
00979 }
00980
00981 void
00982 embPattern_stitchText(EmbPattern *p, EmbRect rect, int thread_index, int style)
00983 {
00984 }
00985
00986 void
00987 embPattern_convertGeometry(EmbPattern* p)
00988 {
00989     int i;
00990     for (i=0; i<p->geometry->count; i++) {
00991         EmbGeometry g = p->geometry->geometry[i];
00992         switch (g.type) {
00993             case EMB_ARC: {
00994                 /* To Do make the thread up here. */
00995                 embPattern_stitchArc(p, g.object.arc, 0, 0);
00996                 break;
00997             }
00998             case EMB_CIRCLE: {
00999                 /* To Do make the thread up here. */
01000                 embPattern_stitchCircle(p, g.object.circle, 0, 0);
01001                 break;
01002             }
01003             case EMB_ELLIPSE: {
01004                 /* To Do make the thread up here. */
01005                 embPattern_stitchEllipse(p, g.object.ellipse, 0, 0);
01006                 break;
01007             }
01008             case EMB_RECT: {
01009                 /* To Do make the thread up here. */
01010                 embPattern_stitchRect(p, g.object.rect, 0, 0);
01011                 break;
01012             }
01013             default:
01014                 break;
01015         }
01016     }
01017     /* Now ignore the geometry when writing. */
01018     p->geometry->count = 0;
01019 }
01020

```

4.13 src/formats.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include "embroidery_internal.h"
```

Functions

- void [safe_free](#) (void *data)
- int [embFormat_getExtension](#) (const char *fileName, char *ending)
- int [emb_identify_format](#) (const char *fileName)
- short [fread_int16](#) (FILE *f)
- unsigned short [fread_uint16](#) (FILE *f)
- int [fread_int32_be](#) (FILE *f)
- void [fpad](#) (FILE *file, char c, int n)
- void [binaryWriteShort](#) (FILE *f, short data)
- void [binaryWriteUShort](#) (FILE *f, unsigned short data)
- void [binaryWriteUShortBE](#) (FILE *f, unsigned short data)
- void [binaryWriteInt](#) (FILE *f, int data)
- void [binaryWriteIntBE](#) (FILE *f, int data)
- void [binaryWriteUInt](#) (FILE *f, unsigned int data)
- void [binaryWriteUIntBE](#) (FILE *f, unsigned int data)
- char [embPattern_read](#) (EmbPattern *pattern, const char *fileName, int format)
- char [embPattern_write](#) (EmbPattern *pattern, const char *fileName, int format)
- char [embPattern_readAuto](#) (EmbPattern *pattern, const char *fileName)
- char [embPattern_writeAuto](#) (EmbPattern *pattern, const char *fileName)

Variables

- EmbFormatList [formatTable](#) [numberOfFormats]
- const char [imageWithFrame](#) [38][48]

4.13.1 Function Documentation

4.13.1.1 [binaryWriteInt\(\)](#)

```
void binaryWriteInt (
    FILE * f,
    int data )
```

Definition at line [235](#) of file [formats.c](#).

4.13.1.2 **binaryWriteIntBE()**

```
void binaryWriteIntBE (
    FILE * f,
    int data )
```

Definition at line [241](#) of file [formats.c](#).

4.13.1.3 **binaryWriteShort()**

```
void binaryWriteShort (
    FILE * f,
    short data )
```

Definition at line [217](#) of file [formats.c](#).

4.13.1.4 **binaryWriteUInt()**

```
void binaryWriteUInt (
    FILE * f,
    unsigned int data )
```

Definition at line [247](#) of file [formats.c](#).

4.13.1.5 **binaryWriteUIntBE()**

```
void binaryWriteUIntBE (
    FILE * f,
    unsigned int data )
```

Definition at line [253](#) of file [formats.c](#).

4.13.1.6 **binaryWriteUShort()**

```
void binaryWriteUShort (
    FILE * f,
    unsigned short data )
```

Definition at line [223](#) of file [formats.c](#).

4.13.1.7 binaryWriteUShortBE()

```
void binaryWriteUShortBE (
    FILE * f,
    unsigned short data )
```

Definition at line [229](#) of file [formats.c](#).

4.13.1.8 emb_identify_format()

```
int emb_identify_format (
    const char * fileName )
```

Definition at line [167](#) of file [formats.c](#).

4.13.1.9 embFormat_getExtension()

```
int embFormat_getExtension (
    const char * fileName,
    char * ending )
```

Definition at line [138](#) of file [formats.c](#).

4.13.1.10 embPattern_read()

```
char embPattern_read (
    EmbPattern * pattern,
    const char * fileName,
    int format )
```

Definition at line [259](#) of file [formats.c](#).

4.13.1.11 embPattern_readAuto()

```
char embPattern_readAuto (
    EmbPattern * pattern,
    const char * fileName )
```

Definition at line [706](#) of file [formats.c](#).

4.13.1.12 **embPattern_write()**

```
char embPattern_write (
    EmbPattern * pattern,
    const char * fileName,
    int format )
```

Definition at line [480](#) of file [formats.c](#).

4.13.1.13 **embPattern_writeAuto()**

```
char embPattern_writeAuto (
    EmbPattern * pattern,
    const char * fileName )
```

Definition at line [718](#) of file [formats.c](#).

4.13.1.14 **fpad()**

```
void fpad (
    FILE * file,
    char c,
    int n )
```

Definition at line [207](#) of file [formats.c](#).

4.13.1.15 **fread_int16()**

```
short fread_int16 (
    FILE * f )
```

Definition at line [183](#) of file [formats.c](#).

4.13.1.16 **fread_int32_be()**

```
int fread_int32_be (
    FILE * f )
```

Definition at line [199](#) of file [formats.c](#).

4.13.1.17 `fread_uint16()`

```
unsigned short fread_uint16 (
    FILE * f )
```

Definition at line 191 of file [formats.c](#).

4.13.1.18 `safe_free()`

```
void safe_free (
    void * data )
```

Definition at line 129 of file [formats.c](#).

4.13.2 Variable Documentation

4.13.2.1 `formatTable`

```
EmbFormatList formatTable[numberOfFormats]
```

Definition at line 21 of file [formats.c](#).

4.13.2.2 `imageWithFrame`

```
const char imageWithFrame[38][48]
```

Definition at line 87 of file [formats.c](#).

4.14 formats.c

[Go to the documentation of this file.](#)

```

00001 /*
00002 * This file is part of libembroidery.
00003 *
00004 * Copyright 2018-2022 The Embroidermodder Team
00005 * Licensed under the terms of the zlib license.
00006 *
00007 * This file contains all the read and write functions for the
00008 * library.
00009 */
00010
00011 #include <stdio.h>
00012 #include <stdlib.h>
00013 #include <string.h>
00014 #include <math.h>
00015 #include <ctype.h>
00016
00017 #include "embroidery_internal.h"
00018
00019 /* TODO: This list needs reviewed in case some stitch
00020   formats also can contain object data (EMBFORMAT_STCHANDOBJ). */
00021 EmbFormatList formatTable[numberOfFormats] = {
00022     {"10o", "Toyota Embroidery Format",           'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00023     {"100", "Toyota Embroidery Format",           'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00024     {"art", "Bernina Embroidery Format",          ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00025     {"bmc", "Bitmap Cache Embroidery Format",    ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00026     {"bro", "Bits & Volts Embroidery Format",   'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00027     {"cnd", "Melco Embroidery Format",            ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00028     {"col", "Embroidery Thread Color Format",    'U', 'U', EMBFORMAT_STITCHONLY, 1, 0, 0},
00029     {"csd", "Singer Embroidery Format",            'U', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00030     {"csv", "Comma Separated Values Format",     'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00031     {"dat", "Barudan Embroidery Format",          'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00032     {"dem", "Melco Embroidery Format",            ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00033     {"dsb", "Barudan Embroidery Format",          'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00034     {"dst", "Tajima Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 1, 0},
00035     {"dsz", "ZSK USA Embroidery Format",          'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00036     {"dxf", "Drawing Exchange Format",           ' ', ' ', EMBFORMAT_OBJECTONLY, 0, 0, 0},
00037     {"edr", "Embird Embroidery Format",          'U', 'U', EMBFORMAT_STITCHONLY, 1, 0, 0},
00038     {"emd", "Elna Embroidery Format",             'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00039     {"exp", "Melco Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 1, 0},
00040     {"exy", "Eltac Embroidery Format",            'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00041     {"eys", "Sierra Expanded Embroidery Format", ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00042     {"fxy", "Fortron Embroidery Format",          'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00043     {"gc", "Smoothie G-Code Format",              ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00044     {"gnc", "Great Notions Embroidery Format",   ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00045     {"gt", "Gold Thread Embroidery Format",       'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00046     {"hus", "Husqvarna Viking Embroidery Format", 'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00047     {"inb", "Inbro Embroidery Format",             'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00048     {"inf", "Embroidery Color Format",            'U', 'U', EMBFORMAT_STITCHONLY, 1, 0, 0},
00049     {"jef", "Janome Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00050     {"ksm", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00051     {"max", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00052     {"mit", "Mitsubishi Embroidery Format",      'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00053     {"new", "Ameco Embroidery Format",            'U', ' ', EMBFORMAT_STITCHONLY, 0, 1, 0},
00054     {"ofm", "Melco Embroidery Format",            'U', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00055     {"pcd", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00056     {"pcm", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00057     {"pcq", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00058     {"pcs", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00059     {"pec", "Brother Embroidery Format",          'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00060     {"pel", "Brother Embroidery Format",          ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00061     {"pem", "Brother Embroidery Format",          ' ', ' ', EMBFORMAT_STITCHONLY, 0, 0, 0},
00062     {"pes", "Brother Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00063     {"phb", "Brother Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00064     {"phc", "Brother Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00065     {"plt", "AutoCAD Plot Drawing Format",       'U', 'U', EMBFORMAT_STITCHONLY, 0, 1, 0},
00066     {"rgb", "RGB Embroidery Format",              'U', 'U', EMBFORMAT_STITCHONLY, 1, 0, 0},
00067     {"sew", "Janome Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00068     {"shv", "Husqvarna Viking Embroidery Format", 'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00069     {"sst", "Sunstar Embroidery Format",          'U', 'U', EMBFORMAT_STITCHONLY, 0, 1, 0},
00070     {"stx", "Data Stitch Embroidery Format",     'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00071     {"svg", "Scalable Vector Graphics",           'U', 'U', EMBFORMAT_OBJECTONLY, 0, 0, 0},
00072     {"t01", "Pfaff Embroidery Format",             'U', 'U', EMBFORMAT_STITCHONLY, 0, 1, 0},
00073     {"t09", "Pfaff Embroidery Format",             'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00074     {"tap", "Happy Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 1, 0},
00075     {"thr", "ThredWorks Embroidery Format",      'U', 'U', EMBFORMAT_STITCHONLY, 1, 0, 0},
00076     {"txt", "Text File",                          ' ', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00077     {"u00", "Barudan Embroidery Format",          'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00078     {"u01", "Barudan Embroidery Format",          ' ', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00079     {"vip", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00080     {"vp3", "Pfaff Embroidery Format",            'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00081     {"xxx", "Singer Embroidery Format",           'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0},
00082     {"zsk", "ZSK USA Embroidery Format",          'U', 'U', EMBFORMAT_STITCHONLY, 0, 0, 0}

```



```
00170     char ending[5];
00171     if (!embFormat_getExtension(fileName, ending)) {
00172         return 0;
00173     }
00174     for (i = 0; i < numberOfFormats; i++) {
00175         if (!strcmp(ending, formatTable[i].extension)) {
00176             return i;
00177         }
00178     }
00179     return -1;
00180 }
00181
00182 short
00183 fread_int16(FILE* f)
00184 {
00185     short x;
00186     embInt_read(f, "fread_int16", &x, EMB_INT16_LITTLE);
00187     return x;
00188 }
00189
00190 unsigned short
00191 fread_uint16(FILE* f)
00192 {
00193     unsigned short x;
00194     embInt_read(f, "fread_uint16", &x, EMB_INT16_LITTLE);
00195     return x;
00196 }
00197
00198 int
00199 fread_int32_be(FILE* f)
00200 {
00201     int x;
00202     embInt_read(f, "fread_int32_be", &x, EMB_INT32_BIG);
00203     return x;
00204 }
00205
00206 void
00207 fpad(FILE* file, char c, int n)
00208 {
00209     int i;
00210     for (i = 0; i < n; i++) {
00211         fwrite(&c, 1, 1, file);
00212     }
00213 }
00214
00215
00216 void
00217 binaryWriteShort(FILE* f, short data)
00218 {
00219     embInt_write(f, "binaryWriteShort", &data, EMB_INT16_LITTLE);
00220 }
00221
00222 void
00223 binaryWriteUShort(FILE* f, unsigned short data)
00224 {
00225     embInt_write(f, "binaryWriteUShort", &data, EMB_INT16_LITTLE);
00226 }
00227
00228 void
00229 binaryWriteUShortBE(FILE* f, unsigned short data)
00230 {
00231     embInt_write(f, "binaryWriteUShortBE", &data, EMB_INT16_BIG);
00232 }
00233
00234 void
00235 binaryWriteInt(FILE* f, int data)
00236 {
00237     embInt_write(f, "binaryWriteInt", &data, EMB_INT32_LITTLE);
00238 }
00239
00240 void
00241 binaryWriteIntBE(FILE* f, int data)
00242 {
00243     embInt_write(f, "binaryWriteIntBE", &data, EMB_INT32_BIG);
00244 }
00245
00246 void
00247 binaryWriteUInt(FILE* f, unsigned int data)
00248 {
00249     embInt_write(f, "binaryWriteUInt", &data, EMB_INT32_LITTLE);
00250 }
00251
00252 void
00253 binaryWriteUIntBE(FILE* f, unsigned int data)
00254 {
00255     embInt_write(f, "binaryWriteUIntBE", &data, EMB_INT32_BIG);
00256 }
```

```
00257
00258 char
00259 embPattern_read(EmbPattern* pattern, const char *fileName, int format)
00260 {
00261     int result;
00262     FILE *file;
00263     result = 0;
00264     if (!pattern) {
00265         printf("ERROR: embPattern_read(), pattern argument is null.\n");
00266         return 0;
00267     }
00268     if (!fileName) {
00269         printf("ERROR: embPattern_read(), fileName argument is null.\n");
00270         return 0;
00271     }
00272     file = fopen(fileName, "rb");
00273     if (!file) {
00274         if ((format != EMB_FORMAT_EDR) &&
00275             (format != EMB_FORMAT_RGB) &&
00276             (format != EMB_FORMAT_COL) &&
00277             (format != EMB_FORMAT_INF)) {
00278             printf("ERROR: Failed to open file with name: %s.\n", fileName);
00279         }
00280         return 0;
00281     }
00282     if (formatTable[format].check_for_color_file) {
00283         embPattern_loadExternalColorFile(pattern, fileName);
00284     }
00285     switch (format) {
00286     case EMB_FORMAT_100:
00287         result = read100(pattern, file);
00288         break;
00289     case EMB_FORMAT_100:
00290         result = read100(pattern, file);
00291         break;
00292     case EMB_FORMAT_ART:
00293         result = readArt(pattern, file);
00294         break;
00295     case EMB_FORMAT_BMC:
00296         result = readBmc(pattern, file);
00297         break;
00298     case EMB_FORMAT_BRO:
00299         result = readBro(pattern, file);
00300         break;
00301     case EMB_FORMAT_CND:
00302         result = readCnd(pattern, file);
00303         break;
00304     case EMB_FORMAT_COL:
00305         result = readCol(pattern, file);
00306         break;
00307     case EMB_FORMAT_CSD:
00308         result = readCsd(pattern, file);
00309         break;
00310     case EMB_FORMAT_CSV:
00311         result = readCsv(pattern, file);
00312         break;
00313     case EMB_FORMAT_DAT:
00314         result = readDat(pattern, file);
00315         break;
00316     case EMB_FORMAT_DEM:
00317         result = readDem(pattern, file);
00318         break;
00319     case EMB_FORMAT_DSB:
00320         result = readDsb(pattern, file);
00321         break;
00322     case EMB_FORMAT_DST:
00323         result = readDst(pattern, file);
00324         break;
00325     case EMB_FORMAT_DSZ:
00326         result = readDsz(pattern, file);
00327         break;
00328     case EMB_FORMAT_DXF:
00329         result = readDxf(pattern, file);
00330         break;
00331     case EMB_FORMAT_EDR:
00332         result = readEdr(pattern, file);
00333         break;
00334     case EMB_FORMAT_EMD:
00335         result = readEmd(pattern, file);
00336         break;
00337     case EMB_FORMAT_EXP:
00338         result = readExp(pattern, file);
00339         break;
00340     case EMB_FORMAT_EXY:
00341         result = readExy(pattern, file);
00342         break;
00343     case EMB_FORMAT_EYS:
```

```
00344     result = readEys(pattern, file);
00345     break;
00346 case EMB_FORMAT_FXY:
00347     result = readFxy(pattern, file);
00348     break;
00349 case EMB_FORMAT_GC:
00350     result = readGc(pattern, file);
00351     break;
00352 case EMB_FORMAT_GNC:
00353     result = readGnc(pattern, file);
00354     break;
00355 case EMB_FORMAT_GT:
00356     result = readGt(pattern, file);
00357     break;
00358 case EMB_FORMAT_HUS:
00359     result = readHus(pattern, file);
00360     break;
00361 case EMB_FORMAT_INB:
00362     result = readInb(pattern, file);
00363     break;
00364 case EMB_FORMAT_INF:
00365     result = readInf(pattern, file);
00366     break;
00367 case EMB_FORMAT_JEF:
00368     result = readJef(pattern, file);
00369     break;
00370 case EMB_FORMAT_KSM:
00371     result = readKsm(pattern, file);
00372     break;
00373 case EMB_FORMAT_MAX:
00374     result = readMax(pattern, file);
00375     break;
00376 case EMB_FORMAT_MIT:
00377     result = readMit(pattern, file);
00378     break;
00379 case EMB_FORMAT_NEW:
00380     result = readNew(pattern, file);
00381     break;
00382 case EMB_FORMAT_OFM:
00383     result = readOfm(pattern, file);
00384     break;
00385 case EMB_FORMAT_PCD:
00386     result = readPcd(pattern, fileName, file);
00387     break;
00388 case EMB_FORMAT_PCM:
00389     result = readPcm(pattern, file);
00390     break;
00391 case EMB_FORMAT_PCQ:
00392     result = readPcq(pattern, fileName, file);
00393     break;
00394 case EMB_FORMAT_PCS:
00395     result = readPcs(pattern, fileName, file);
00396     break;
00397 case EMB_FORMAT_PEC:
00398     result = readPec(pattern, fileName, file);
00399     break;
00400 case EMB_FORMAT_PEL:
00401     result = readPel(pattern, file);
00402     break;
00403 case EMB_FORMAT_PEM:
00404     result = readPem(pattern, file);
00405     break;
00406 case EMB_FORMAT_PES:
00407     result = readPes(pattern, fileName, file);
00408     break;
00409 case EMB_FORMAT_PHP:
00410     result = readPhb(pattern, file);
00411     break;
00412 case EMB_FORMAT_PHC:
00413     result = readPhc(pattern, file);
00414     break;
00415 case EMB_FORMAT_PLT:
00416     result = readPlt(pattern, file);
00417     break;
00418 case EMB_FORMAT_RGB:
00419     result = readRgb(pattern, file);
00420     break;
00421 case EMB_FORMAT_SEW:
00422     result = readSew(pattern, file);
00423     break;
00424 case EMB_FORMAT_SHV:
00425     result = readShv(pattern, file);
00426     break;
00427 case EMB_FORMAT_SST:
00428     result = readSst(pattern, file);
00429     break;
00430 case EMB_FORMAT_STX:
```

```
00431     result = readStx(pattern, file);
00432     break;
00433 case EMB_FORMAT_SVG:
00434     result = readSvg(pattern, file);
00435     break;
00436 case EMB_FORMAT_T01:
00437     result = readT01(pattern, file);
00438     break;
00439 case EMB_FORMAT_T09:
00440     result = readT09(pattern, file);
00441     break;
00442 case EMB_FORMAT_TAP:
00443     result = readTap(pattern, file);
00444     break;
00445 case EMB_FORMAT_THR:
00446     result = readThr(pattern, file);
00447     break;
00448 case EMB_FORMAT_TXT:
00449     result = readTxt(pattern, file);
00450     break;
00451 case EMB_FORMAT_U00:
00452     result = readU00(pattern, file);
00453     break;
00454 case EMB_FORMAT_U01:
00455     result = readU01(pattern, file);
00456     break;
00457 case EMB_FORMAT_VIP:
00458     result = readVip(pattern, file);
00459     break;
00460 case EMB_FORMAT_vp3:
00461     result = readVp3(pattern, file);
00462     break;
00463 case EMB_FORMAT_XXX:
00464     result = readXxx(pattern, file);
00465     break;
00466 case EMB_FORMAT_ZSK:
00467     result = readZsk(pattern, file);
00468     break;
00469 default:
00470     break;
00471 }
00472 fclose(file);
00473 if (!formatTable[format].color_only) {
00474     embPattern_end(pattern);
00475 }
00476 return result;
00477 }
00478
00479 char
00480 embPattern_write(EmbPattern* pattern, const char *fileName, int format)
00481 {
00482     FILE *file;
00483     int result = 0;
00484     if (!pattern) {
00485         printf("ERROR: embPattern_write(), pattern argument is null\n");
00486         return 0;
00487     }
00488     if (!fileName) {
00489         printf("ERROR: embPattern_write(), fileName argument is null\n");
00490         return 0;
00491     }
00492     if (pattern->stitch_list->count == 0) {
00493         printf("ERROR: embPattern_write(), pattern contains no stitches\n");
00494         return 0;
00495     }
00496     if (!formatTable[format].color_only) {
00497         embPattern_end(pattern);
00498     }
00499
00500     file = fopen(fileName, "wb");
00501     if (!file) {
00502         printf("Failed to open file with name: %s.", fileName);
00503         return 0;
00504     }
00505     switch (format) {
00506     case EMB_FORMAT_100:
00507         result = write100(pattern, file);
00508         break;
00509     case EMB_FORMAT_100:
00510         result = write10o(pattern, file);
00511         break;
00512     case EMB_FORMAT_ART:
00513         result = writeArt(pattern, file);
00514         break;
00515     case EMB_FORMAT_BMC:
00516         result = writeBmc(pattern, file);
00517         break;
```

```
00518     case EMB_FORMAT_BRO:
00519         result = writeBro(pattern, file);
00520         break;
00521     case EMB_FORMAT_CND:
00522         result = writeCnd(pattern, file);
00523         break;
00524     case EMB_FORMAT_COL:
00525         result = writeCol(pattern, file);
00526         break;
00527     case EMB_FORMAT_CSD:
00528         result = writeCsd(pattern, file);
00529         break;
00530     case EMB_FORMAT_CSV:
00531         result = writeCsv(pattern, file);
00532         break;
00533     case EMB_FORMAT_DAT:
00534         result = writeDat(pattern, file);
00535         break;
00536     case EMB_FORMAT_DEM:
00537         result = writeDem(pattern, file);
00538         break;
00539     case EMB_FORMAT_DSB:
00540         result = writeDsb(pattern, file);
00541         break;
00542     case EMB_FORMAT_DST:
00543         result = writeDst(pattern, file);
00544         break;
00545     case EMB_FORMAT_DSZ:
00546         result = writeDsz(pattern, file);
00547         break;
00548     case EMB_FORMAT_DXF:
00549         result = writeDxf(pattern, file);
00550         break;
00551     case EMB_FORMAT_EDR:
00552         result = writeEdr(pattern, file);
00553         break;
00554     case EMB_FORMAT_EMD:
00555         result = writeEmd(pattern, file);
00556         break;
00557     case EMB_FORMAT_EXP:
00558         result = writeExp(pattern, file);
00559         break;
00560     case EMB_FORMAT_EXY:
00561         result = writeExy(pattern, file);
00562         break;
00563     case EMB_FORMAT_EYS:
00564         result = writeEys(pattern, file);
00565         break;
00566     case EMB_FORMAT_FXY:
00567         result = writeFxy(pattern, file);
00568         break;
00569     case EMB_FORMAT_GC:
00570         result = writeGc(pattern, file);
00571         break;
00572     case EMB_FORMAT_GNC:
00573         result = writeGnc(pattern, file);
00574         break;
00575     case EMB_FORMAT_GT:
00576         result = writeGt(pattern, file);
00577         break;
00578     case EMB_FORMAT_HUS:
00579         result = writeHus(pattern, file);
00580         break;
00581     case EMB_FORMAT_INB:
00582         result = writeInb(pattern, file);
00583         break;
00584     case EMB_FORMAT_INF:
00585         result = writeInf(pattern, file);
00586         break;
00587     case EMB_FORMAT_JEF:
00588         result = writeJef(pattern, file);
00589         break;
00590     case EMB_FORMAT_KSM:
00591         result = writeKsm(pattern, file);
00592         break;
00593     case EMB_FORMAT_MAX:
00594         result = writeMax(pattern, file);
00595         break;
00596     case EMB_FORMAT_MIT:
00597         result = writeMit(pattern, file);
00598         break;
00599     case EMB_FORMAT_NEW:
00600         result = writeNew(pattern, file);
00601         break;
00602     case EMB_FORMAT_OFM:
00603         result = writeOfm(pattern, file);
00604         break;
```

```
00605     case EMB_FORMAT_PCD:
00606         result = writePcd(pattern, file);
00607         break;
00608     case EMB_FORMAT_PCM:
00609         result = writePcm(pattern, file);
00610         break;
00611     case EMB_FORMAT_PCQ:
00612         result = writePcq(pattern, file);
00613         break;
00614     case EMB_FORMAT_PCS:
00615         result = writePcs(pattern, file);
00616         break;
00617     case EMB_FORMAT_PEC:
00618         result = writePec(pattern, fileName, file);
00619         break;
00620     case EMB_FORMAT_PEL:
00621         result = writePel(pattern, file);
00622         break;
00623     case EMB_FORMAT_PEM:
00624         result = writePem(pattern, file);
00625         break;
00626     case EMB_FORMAT_PES:
00627         result = writePes(pattern, fileName, file);
00628         break;
00629     case EMB_FORMAT_PHB:
00630         result = writePhb(pattern, file);
00631         break;
00632     case EMB_FORMAT_PHC:
00633         result = writePhc(pattern, file);
00634         break;
00635     case EMB_FORMAT_PLT:
00636         result = writePlt(pattern, file);
00637         break;
00638     case EMB_FORMAT_RGB:
00639         result = writeRgb(pattern, file);
00640         break;
00641     case EMB_FORMAT_SEW:
00642         result = writeSew(pattern, file);
00643         break;
00644     case EMB_FORMAT_SHV:
00645         result = writeShv(pattern, file);
00646         break;
00647     case EMB_FORMAT_SST:
00648         result = writeSst(pattern, file);
00649         break;
00650     case EMB_FORMAT_STX:
00651         result = writeStx(pattern, file);
00652         break;
00653     case EMB_FORMAT_SVG:
00654         result = writeSvg(pattern, file);
00655         break;
00656     case EMB_FORMAT_T01:
00657         result = writeT01(pattern, file);
00658         break;
00659     case EMB_FORMAT_T09:
00660         result = writeT09(pattern, file);
00661         break;
00662     case EMB_FORMAT_TAP:
00663         result = writeTap(pattern, file);
00664         break;
00665     case EMB_FORMAT_THR:
00666         result = writeThr(pattern, file);
00667         break;
00668     case EMB_FORMAT_TXT:
00669         result = writeTxt(pattern, file);
00670         break;
00671     case EMB_FORMAT_U00:
00672         result = writeU00(pattern, file);
00673         break;
00674     case EMB_FORMAT_U01:
00675         result = writeU01(pattern, file);
00676         break;
00677     case EMB_FORMAT_VIP:
00678         result = writeVip(pattern, file);
00679         break;
00680     case EMB_FORMAT_VP3:
00681         result = writeVp3(pattern, file);
00682         break;
00683     case EMB_FORMAT_XXX:
00684         result = writeXxx(pattern, file);
00685         break;
00686     case EMB_FORMAT_ZSK:
00687         result = writeZsk(pattern, file);
00688         break;
00689     default:
00690         break;
00691 }
```

```

00692     if (formatTable[format].write_external_color_file) {
00693         char externalFileName[1000];
00694         int stub_length;
00695         strcpy(externalFileName, fileName);
00696         stub_length = strlen(fileName)-strlen(formatTable[format].extension);
00697         externalFileName[stub_length] = 0;
00698         strcat(externalFileName, ".rgb");
00699         embPattern_write(pattern, externalFileName, EMB_FORMAT_RGB);
00700     }
00701     fclose(file);
00702     return result;
00703 }
00704
00705 char
00706 embPattern_readAuto(EmbPattern* pattern, const char* fileName)
00707 {
00708     int format = emb_identify_format(fileName);
00709     if (format < 0) {
00710         printf("ERROR: convert(), unsupported read file type: %s\n", fileName);
00711         embPattern_free(pattern);
00712         return 0;
00713     }
00714     return embPattern_read(pattern, fileName, format);
00715 }
00716
00717 char
00718 embPattern_writeAuto(EmbPattern* pattern, const char* fileName)
00719 {
00720     int format = emb_identify_format(fileName);
00721     if (format < 0) {
00722         printf("ERROR: convert(), unsupported write file type: %s\n", fileName);
00723         embPattern_free(pattern);
00724         return 0;
00725     }
00726     return embPattern_write(pattern, fileName, format);
00727 }

```

4.15 src/geometry.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "embroidery.h"

```

Functions

- `EmbGeometry * embGeometry_init (int type_in)`
- `void embGeometry_free (EmbGeometry *obj)`
- `void embGeometry_move (EmbGeometry *obj, EmbVector delta)`
- `EmbRect embGeometry_boundingRect (EmbGeometry *obj)`
- `void embGeometry_vulcanize (EmbGeometry *obj)`

4.15.1 Function Documentation

4.15.1.1 embGeometry_boundingRect()

```

EmbRect embGeometry_boundingRect (
    EmbGeometry * obj )

```

Definition at line 102 of file [geometry.c](#).

4.15.1.2 embGeometry_free()

```
void embGeometry_free (
    EmbGeometry * obj )
```

Definition at line 63 of file [geometry.c](#).

4.15.1.3 embGeometry_init()

```
EmbGeometry * embGeometry_init (
    int type_in )
```

Definition at line 20 of file [geometry.c](#).

4.15.1.4 embGeometry_move()

```
void embGeometry_move (
    EmbGeometry * obj,
    EmbVector delta )
```

Definition at line 81 of file [geometry.c](#).

4.15.1.5 embGeometry_vulcanize()

```
void embGeometry_vulcanize (
    EmbGeometry * obj )
```

Definition at line 128 of file [geometry.c](#).

4.16 geometry.c

[Go to the documentation of this file.](#)

```
00001 /*
00002  * GEOMETRY
00003  * This file is part of libembroidery.
00004  *
00005  * Copyright 2018-2023 The Embroidermodder Team
00006  * Licensed under the terms of the zlib license.
00007 */
00008
00009 #include <stdio.h>
00010 #include <stdlib.h>
00011 #include <math.h>
00012
00013 #include "embroidery.h"
00014
00015
00016 /* Our generic object interface backends to each individual type,
00017 */
00018 */
```

```

00019 EmbGeometry *
00020 embGeometry_init(int type_in)
00021 {
00022     EmbGeometry *obj = (EmbGeometry*)malloc(sizeof(EmbGeometry));
00023     obj->type = type_in;
00024     obj->color.r = 0;
00025     obj->color.g = 0;
00026     obj->color.b = 0;
00027
00028     /*
00029     // QGraphicsItem* parent
00030     debug_message("BaseObject Constructor()");
00031
00032     objPen.setCapStyle(RoundCap);
00033     objPen.setJoinStyle(RoundJoin);
00034     lwtPen.setCapStyle(RoundCap);
00035     lwtPen.setJoinStyle(RoundJoin);
00036
00037     objID = QDateTime::currentMSecsSinceEpoch();
00038 */
00039
00040     switch (obj->type) {
00041         case EMB_ARC: {
00042             obj->object.arc = embArc_init();
00043             /*
00044             embArc_init(EmbArc arc_in, unsigned int rgb, int lineType)
00045             arc = arc_in;
00046
00047             setFlag(ItemIsSelectable, true);
00048
00049             calculateArcData(arc);
00050
00051             setColor(rgb);
00052             setLineType(lineType);
00053             setLineWeight(0.35); //TODO: pass in proper linewidth
00054             setPen(objPen);
00055             */
00056             break;
00057         }
00058     }
00059     return obj;
00060 }
00061
00062 void
00063 embGeometry_free(EmbGeometry *obj)
00064 {
00065     switch (obj->type) {
00066         case EMB_ARC: {
00067             return;
00068         }
00069         case EMB_CIRCLE: {
00070             return;
00071         }
00072         case EMB_ELLIPSE: {
00073             return;
00074         }
00075         default:
00076             return;
00077     }
00078 }
00079
00080 void
00081 embGeometry_move(EmbGeometry *obj, EmbVector delta)
00082 {
00083     switch (obj->type) {
00084         case EMB_ARC: {
00085             EmbArc *arc = &(obj->object.arc);
00086             arc->start = embVector_add(arc->start, delta);
00087             arc->mid = embVector_add(arc->mid, delta);
00088             arc->end = embVector_add(arc->end, delta);
00089             return;
00090         }
00091         case EMB_CIRCLE: {
00092             EmbCircle *circle = &(obj->object.circle);
00093             circle->center = embVector_add(circle->center, delta);
00094             return;
00095         }
00096         default:
00097             break;
00098     }
00099 }
00100
00101 EmbRect
00102 embGeometry_boundingRect(EmbGeometry *obj)
00103 {
00104     EmbRect r;
00105     if ((obj->type == EMB_ARC)) {

```

```

00106      /*
00107      arcRect.setWidth(radius*2.0);
00108      arcRect.setHeight(radius*2.0);
00109      arcRect.moveCenter(EmbVector(0,0));
00110      setRect(arcRect);
00111      */
00112  }
00113  r.top = 1.0;
00114  r.bottom = 1.0;
00115  r.left = 1.0;
00116  r.right = 1.0;
00117  /*
00118  "Base"
00119  //If gripped, force this object to be drawn even if it is offscreen
00120  if (objectRubberMode() == OBJ_RUBBER_GRIP)
00121      return scene()->sceneRect();
00122  return path().boundingRect();
00123  */
00124  return r;
00125 }
00126
00127 void
00128 embGeometry_vulcanize(EmbGeometry *obj)
00129 {
00130     switch (obj->type) {
00131     case EMB_ARC:
00132     case EMB_CIRCLE:
00133     case EMB_DIM_LEADER:
00134     case EMB_ELLIPSE:
00135     case EMB_IMAGE:
00136     case EMB_LINE:
00137     case EMB_POINT:
00138     case EMB_RECT:
00139     case EMB_TEXT_SINGLE:
00140         /*
00141         updateRubber();
00142
00143         setRubberMode(OBJ_RUBBER_OFF);
00144         */
00145         break;
00146     default:
00147         break;
00148     }
00149     if (obj->type == EMB_PATH) {
00150         /*
00151         updateRubber();
00152
00153         setRubberMode(OBJ_RUBBER_OFF);
00154
00155         if (!normalPath.elementCount())
00156             critical_messagebox(0, translate("Empty Path Error"), translate("The path added contains
no points. The command that created this object has flawed logic."));
00157         */
00158     }
00159     if (obj->type == EMB_POLYGON) {
00160         /*
00161         updateRubber();
00162
00163         setRubberMode(OBJ_RUBBER_OFF);
00164
00165         if (!normalPath.elementCount())
00166             critical_messagebox(0, translate("Empty Polygon Error"), translate("The polygon added
contains no points. The command that created this object has flawed logic."));
00167         */
00168     }
00169     if (obj->type == EMB_POLYLINE) {
00170         /*
00171         updateRubber();
00172
00173         setRubberMode(OBJ_RUBBER_OFF);
00174
00175         if (!normalPath.elementCount())
00176             critical_messagebox(0, translate("Empty embPolyline Error"), translate("The embPolyline
added contains no points. The command that created this object has flawed logic."));
00177         */
00178     }
00179 }

```

4.17 src/image.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <math.h>
#include "embroidery_internal.h"
```

Functions

- void [writeImage](#) (FILE *file, unsigned char image[][48])
- float [image_diff](#) (unsigned char *a, unsigned char *b, int size)

4.17.1 Function Documentation

4.17.1.1 [image_diff\(\)](#)

```
float image_diff (
    unsigned char * a,
    unsigned char * b,
    int size )
```

Definition at line 54 of file [image.c](#).

4.17.1.2 [writeImage\(\)](#)

```
void writeImage (
    FILE * file,
    unsigned char image[][48] )
```

Definition at line 26 of file [image.c](#).

4.18 [image.c](#)

[Go to the documentation of this file.](#)

```
00001 /*
00002  * This file is part of libembroidery.
00003  *
00004  * Copyright 2018-2023 The Embroidermodder Team
00005  * Licensed under the terms of the zlib license.
00006  *
00007  * This file contains all the read and write functions for the
00008  * library.
00009  *
00010 ****
00011 *
00012 * This backends to the stb libraries and nanosvg library.
00013 *
00014 * Use Python PEP7 for coding style.
00015 */
00016
00017 #include <stdio.h>
00018 #include <stdlib.h>
00019 #include <math.h>
00020
00021 #include "embroidery_internal.h"
```

```
00022
00023
00024 /* for the PES embedded */
00025 void
00026 writeImage(FILE* file, unsigned char image[][48])
00027 {
00028     int i, j;
00029
00030     if (!file) {
00031         printf("ERROR: format-pec.c writeImage(), file argument is null\n");
00032         return;
00033     }
00034     for (i = 0; i < 38; i++) {
00035         for (j = 0; j < 6; j++) {
00036             int offset = j * 8;
00037             unsigned char output = 0;
00038             output |= (unsigned char)(image[i][offset] != 0);
00039             output |= (unsigned char)(image[i][offset + 1] != (unsigned char)0) << 1;
00040             output |= (unsigned char)(image[i][offset + 2] != (unsigned char)0) << 2;
00041             output |= (unsigned char)(image[i][offset + 3] != (unsigned char)0) << 3;
00042             output |= (unsigned char)(image[i][offset + 4] != (unsigned char)0) << 4;
00043             output |= (unsigned char)(image[i][offset + 5] != (unsigned char)0) << 5;
00044             output |= (unsigned char)(image[i][offset + 6] != (unsigned char)0) << 6;
00045             output |= (unsigned char)(image[i][offset + 7] != (unsigned char)0) << 7;
00046             fwrite(&output, 1, 1, file);
00047         }
00048     }
00049 }
00050
00051 /*
00052 *
00053 */
00054 float image_diff(unsigned char *a, unsigned char *b, int size)
00055 {
00056     int i;
00057     float total = 0.0;
00058     for (i=0; i<size; i++) {
00059         int diff = a[i] - b[i];
00060         total += diff*diff;
00061     }
00062     return total;
00063 }
00064
00065 #ifdef LIBEMBROIDERY_CLI
00066 #define STB_IMAGE_IMPLEMENTATION
00067 #include "stb/stb_image.h"
00068
00069 #define STB_IMAGE_WRITE_IMPLEMENTATION
00070 #include "stb/stb_image_write.h"
00071
00072 #define NANOSVG_ALL_COLOR_KEYWORDS
00073 #define NANOSVG_IMPLEMENTATION
00074 #include "nanosvg/src/nanosvg.h"
00075
00076 #define NANOSVGRAST_IMPLEMENTATION
00077 #include "nanosvg/src/nanosvgrast.h"
00078
00079
00080 /* Basic Render
00081 * -----
00082 * Backends rendering to nanosvg/stb_image.
00083 *
00084 * The caller is responsible for the memory in p.
00085 */
00086 int
00087 embPattern_render(EmbPattern *p, char *fname)
00088 {
00089     const char *tmp_fname = "libembroidery_temp.svg";
00090     NSVGimage *image = NULL;
00091     NSVGGrasterizer rast;
00092     unsigned char *img_data = NULL;
00093     embPattern_writeAuto(p, tmp_fname);
00094     image = nsvgParseFromFile(tmp_fname, "px", 96.0f);
00095     img_data = malloc(4*image->width*image->height);
00096     nsvgRasterize(
00097         &rast,
00098         image,
00099         0, 0, 1,
00100         img_data,
00101         image->width,
00102         image->height,
00103         4*image->width);
00104     stbi_write_png(
00105         fname,
00106         image->width,
00107         image->height,
00108         4,
```

```

00109         img_data,
00110         4*image->width);
00111     return 0;
00112 }
00113
00114 /* Simulate the stitching of a pattern, using the image for rendering
00115 * hints about how to represent the pattern.
00116 */
00117 int
00118 embPattern_simulate(EmbPattern *pattern, char *fname)
00119 {
00120
00121     embPattern_render(pattern, fname);
00122     return 0;
00123 }
00124
00125 EmbImage
00126 embImage_create(int width, int height)
00127 {
00128     EmbImage image;
00129     image.width = width;
00130     image.height = height;
00131     image.data = malloc(4*width*height);
00132     return image;
00133 }
00134 }
00135
00136 void
00137 embImage_read(EmbImage *image, char *fname)
00138 {
00139     int channels_in_file;
00140     image->data = stbi_load(
00141         fname,
00142         &(image->width),
00143         &(image->height),
00144         &channels_in_file,
00145         3);
00146 }
00147
00148 int
00149 embImage_write(EmbImage *image, char *fname)
00150 {
00151     return stbi_write_png(
00152         fname,
00153         image->width,
00154         image->height,
00155         4,
00156         image->data,
00157         4*image->width);
00158 }
00159
00160 void
00161 embImage_free(EmbImage *image)
00162 {
00163     free(image->data);
00164 }
00165 #endif
00166

```

4.19 src/main.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include "embroidery_internal.h"

```

Macros

- #define FLAG_TO 0
- #define FLAG_TO_SHORT 1

- #define FLAG_HELP 2
- #define FLAG_HELP_SHORT 3
- #define FLAG_FORMATS 4
- #define FLAG_FORMATS_SHORT 5
- #define FLAG QUIET 6
- #define FLAG QUIET_SHORT 7
- #define FLAG_VERBOSE 8
- #define FLAG_VERBOSE_SHORT 9
- #define FLAG_VERSION 10
- #define FLAG_VERSION_SHORT 11
- #define FLAG_CIRCLE 12
- #define FLAG_CIRCLE_SHORT 13
- #define FLAG_ELLIPSE 14
- #define FLAG_ELLIPSE_SHORT 15
- #define FLAG_LINE 16
- #define FLAG_LINE_SHORT 17
- #define FLAG_POLYGON 18
- #define FLAG_POLYGON_SHORT 19
- #define FLAG_POLYLINE 20
- #define FLAG_POLYLINE_SHORT 21
- #define FLAG_RENDER 22
- #define FLAG_RENDER_SHORT 23
- #define FLAG_SATIN 24
- #define FLAG_SATIN_SHORT 25
- #define FLAG_STITCH 26
- #define FLAG_STITCH_SHORT 27
- #define FLAG_TEST 28
- #define FLAG_FULL_TEST_SUITE 29
- #define FLAG_HILBERT_CURVE 30
- #define FLAG_SIERPINSKI_TRIANGLE 31
- #define FLAG_FILL 32
- #define FLAG_FILL_SHORT 33
- #define FLAG_SIMULATE 34
- #define FLAG_COMBINE 35
- #define FLAG_CROSS_STITCH 36
- #define NUM_FLAGS 37

Functions

- void embVector_print (EmbVector v, char *label)
- void embArc_print (EmbArc arc)
- int check_header_present (FILE *file, int minimum_header_length)
- unsigned int sectorSize (bcf_file *bcfFile)
- int haveExtraDIFATSectors (bcf_file *file)
- int seekToSector (bcf_file *bcfFile, FILE *file, const unsigned int sector)
- void parseDIFATSectors (FILE *file, bcf_file *bcfFile)
- int bcfFile_read (FILE *file, bcf_file *bcfFile)
- FILE * GetFile (bcf_file *bcfFile, FILE *file, char *fileToFind)
- void bcf_file_free (bcf_file *bcfFile)
- bcf_file_difat * bcf_difat_create (FILE *file, unsigned int fatSectors, const unsigned int sectorSize)
- unsigned int entriesInDifatSector (bcf_file_difat *fat)
- unsigned int readFullSector (FILE *file, bcf_file_difat *bcfFile, unsigned int *difatEntriesToRead)
- void parseDirectoryEntryName (FILE *file, bcf_directory_entry *dir)
- bcf_directory * CompoundFileDialog (const unsigned int maxNumberOfDirectoryEntries)

- `EmbTime parseTime (FILE *file)`
- `bcf_directory_entry * CompoundFileDirectoryEntry (FILE *file)`
- `void readNextSector (FILE *file, bcf_directory *dir)`
- `void bcf_directory_free (bcf_directory **dir)`
- `bcf_file_fat * bcfFileFat_create (const unsigned int sectorSize)`
- `void loadFatFromSector (bcf_file_fat *fat, FILE *file)`
- `bcf_file_header bcfFileHeader_read (FILE *file)`
- `void embSatinOutline_generateSatinOutline (EmbArray *lines, EmbReal thickness, EmbSatinOutline *result)`
- `EmbArray * embSatinOutline_renderStitches (EmbSatinOutline *result, EmbReal density)`
- `void write_24bit (FILE *file, int x)`
- `int embColor_distance (EmbColor a, EmbColor b)`
- `void embColor_read (FILE *f, EmbColor *c, int toRead)`
- `void embColor_write (FILE *f, EmbColor c, int toWrite)`
- `int embThread_findNearestColor (EmbColor color, EmbColor *color_list, int n_colors)`
- `int embThread_findNearestThread (EmbColor color, EmbThread *thread_list, int n_threads)`
- `EmbThread embThread_getRandom (void)`
- `void binaryReadString (FILE *file, char *buffer, int maxLength)`
- `void binaryReadUnicodeString (FILE *file, char *buffer, const int stringLength)`
- `int stringInArray (const char *s, const char **array)`
- `int emb_readline (FILE *file, char *line, int maxLength)`
- `void get_trim_bounds (char const *s, char const **firstWord, char const **trailingSpace)`
- `char * copy_trim (char const *s)`
- `char * emb_optOut (EmbReal num, char *str)`
- `void embTime_initNow (EmbTime *t)`
- `EmbTime embTime_time (EmbTime *t)`

Variables

- `EmbThread black_thread = { { 0, 0, 0 }, "Black", "Black" }`
- `int emb_verbose = 0`
- `int emb_error = 0`
- `const EmbReal embConstantPi = 3.1415926535`
- `const unsigned int difatEntriesInHeader = 109`
- `const unsigned int sizeOfFatEntry = sizeof(unsigned int)`
- `const unsigned int sizeOfDifatEntry = 4`
- `const unsigned int sizeOfChainingEntryAtEndOfDifatSector = 4`
- `const unsigned int sizeOfDirectoryEntry = 128`
- `char const WHITESPACE [] = " \t\n\r"`

4.19.1 Macro Definition Documentation

4.19.1.1 FLAG_CIRCLE

```
#define FLAG_CIRCLE 12
```

Definition at line 29 of file [main.c](#).

4.19.1.2 FLAG_CIRCLE_SHORT

```
#define FLAG_CIRCLE_SHORT 13
```

Definition at line [30](#) of file [main.c](#).

4.19.1.3 FLAG_COMBINE

```
#define FLAG_COMBINE 35
```

Definition at line [52](#) of file [main.c](#).

4.19.1.4 FLAG_CROSS_STITCH

```
#define FLAG_CROSS_STITCH 36
```

Definition at line [53](#) of file [main.c](#).

4.19.1.5 FLAG_ELLIPSE

```
#define FLAG_ELLIPSE 14
```

Definition at line [31](#) of file [main.c](#).

4.19.1.6 FLAG_ELLIPSE_SHORT

```
#define FLAG_ELLIPSE_SHORT 15
```

Definition at line [32](#) of file [main.c](#).

4.19.1.7 FLAG_FILL

```
#define FLAG_FILL 32
```

Definition at line [49](#) of file [main.c](#).

4.19.1.8 FLAG_FILL_SHORT

```
#define FLAG_FILL_SHORT 33
```

Definition at line [50](#) of file [main.c](#).

4.19.1.9 FLAG_FORMATS

```
#define FLAG_FORMATS 4
```

Definition at line [21](#) of file [main.c](#).

4.19.1.10 FLAG_FORMATS_SHORT

```
#define FLAG_FORMATS_SHORT 5
```

Definition at line [22](#) of file [main.c](#).

4.19.1.11 FLAG_FULL_TEST_SUITE

```
#define FLAG_FULL_TEST_SUITE 29
```

Definition at line [46](#) of file [main.c](#).

4.19.1.12 FLAG_HELP

```
#define FLAG_HELP 2
```

Definition at line [19](#) of file [main.c](#).

4.19.1.13 FLAG_HELP_SHORT

```
#define FLAG_HELP_SHORT 3
```

Definition at line [20](#) of file [main.c](#).

4.19.1.14 FLAG_HILBERT_CURVE

```
#define FLAG_HILBERT_CURVE 30
```

Definition at line [47](#) of file [main.c](#).

4.19.1.15 FLAG_LINE

```
#define FLAG_LINE 16
```

Definition at line [33](#) of file [main.c](#).

4.19.1.16 FLAG_LINE_SHORT

```
#define FLAG_LINE_SHORT 17
```

Definition at line [34](#) of file [main.c](#).

4.19.1.17 FLAG_POLYGON

```
#define FLAG_POLYGON 18
```

Definition at line [35](#) of file [main.c](#).

4.19.1.18 FLAG_POLYGON_SHORT

```
#define FLAG_POLYGON_SHORT 19
```

Definition at line [36](#) of file [main.c](#).

4.19.1.19 FLAG_POLYLINE

```
#define FLAG_POLYLINE 20
```

Definition at line [37](#) of file [main.c](#).

4.19.1.20 FLAG_POLYLINE_SHORT

```
#define FLAG_POLYLINE_SHORT 21
```

Definition at line [38](#) of file [main.c](#).

4.19.1.21 FLAG_QUIET

```
#define FLAG_QUIET 6
```

Definition at line [23](#) of file [main.c](#).

4.19.1.22 FLAG_QUIET_SHORT

```
#define FLAG_QUIET_SHORT 7
```

Definition at line [24](#) of file [main.c](#).

4.19.1.23 FLAG_RENDER

```
#define FLAG_RENDER 22
```

Definition at line [39](#) of file [main.c](#).

4.19.1.24 FLAG_RENDER_SHORT

```
#define FLAG_RENDER_SHORT 23
```

Definition at line [40](#) of file [main.c](#).

4.19.1.25 FLAG_SATIN

```
#define FLAG_SATIN 24
```

Definition at line [41](#) of file [main.c](#).

4.19.1.26 FLAG_SATIN_SHORT

```
#define FLAG_SATIN_SHORT 25
```

Definition at line [42](#) of file [main.c](#).

4.19.1.27 FLAG_SIERPINSKI_TRIANGLE

```
#define FLAG_SIERPINSKI_TRIANGLE 31
```

Definition at line [48](#) of file [main.c](#).

4.19.1.28 FLAG_SIMULATE

```
#define FLAG_SIMULATE 34
```

Definition at line [51](#) of file [main.c](#).

4.19.1.29 FLAG_STITCH

```
#define FLAG_STITCH 26
```

Definition at line [43](#) of file [main.c](#).

4.19.1.30 FLAG_STITCH_SHORT

```
#define FLAG_STITCH_SHORT 27
```

Definition at line [44](#) of file [main.c](#).

4.19.1.31 FLAG_TEST

```
#define FLAG_TEST 28
```

Definition at line [45](#) of file [main.c](#).

4.19.1.32 FLAG_TO

```
#define FLAG_TO 0
```

Definition at line [17](#) of file [main.c](#).

4.19.1.33 FLAG_TO_SHORT

```
#define FLAG_TO_SHORT 1
```

Definition at line [18](#) of file [main.c](#).

4.19.1.34 FLAG_VERBOSE

```
#define FLAG_VERBOSE 8
```

Definition at line [25](#) of file [main.c](#).

4.19.1.35 FLAG_VERBOSE_SHORT

```
#define FLAG_VERBOSE_SHORT 9
```

Definition at line [26](#) of file [main.c](#).

4.19.1.36 FLAG_VERSION

```
#define FLAG_VERSION 10
```

Definition at line [27](#) of file [main.c](#).

4.19.1.37 FLAG_VERSION_SHORT

```
#define FLAG_VERSION_SHORT 11
```

Definition at line [28](#) of file [main.c](#).

4.19.1.38 NUM_FLAGS

```
#define NUM_FLAGS 37
```

Definition at line 54 of file [main.c](#).

4.19.2 Function Documentation

4.19.2.1 bcf_difat_create()

```
bcf_file_difat * bcf_difat_create (
    FILE * file,
    unsigned int fatSectors,
    const unsigned int sectorSize )
```

Definition at line 276 of file [main.c](#).

4.19.2.2 bcf_directory_free()

```
void bcf_directory_free (
    bcf_directory ** dir )
```

Definition at line 448 of file [main.c](#).

4.19.2.3 bcf_file_free()

```
void bcf_file_free (
    bcf_file * bcfFile )
```

Definition at line 267 of file [main.c](#).

4.19.2.4 bcfFile_read()

```
int bcfFile_read (
    FILE * file,
    bcf_file * bcfFile )
```

Definition at line 196 of file [main.c](#).

4.19.2.5 bcfFileFat_create()

```
bcf_file_fat * bcfFileFat_create (
    const unsigned int sectorSize )
```

Definition at line [467](#) of file [main.c](#).

4.19.2.6 bcfFileHeader_read()

```
bcf_file_header bcfFileHeader_read (
    FILE * file )
```

Definition at line [495](#) of file [main.c](#).

4.19.2.7 binaryReadString()

```
void binaryReadString (
    FILE * file,
    char * buffer,
    int maxLength )
```

Definition at line [766](#) of file [main.c](#).

4.19.2.8 binaryReadUnicodeString()

```
void binaryReadUnicodeString (
    FILE * file,
    char * buffer,
    const int stringLength )
```

Definition at line [777](#) of file [main.c](#).

4.19.2.9 check_header_present()

```
int check_header_present (
    FILE * file,
    int minimum_header_length )
```

Definition at line [149](#) of file [main.c](#).

4.19.2.10 CompoundFileDirectory()

```
bcf_directory * CompoundFileDirectory (
    const unsigned int maxNumberOfDirectoryEntries )
```

Definition at line 360 of file [main.c](#).

4.19.2.11 CompoundFileDirectoryEntry()

```
bcf_directory_entry * CompoundFileDirectoryEntry (
    FILE * file )
```

Definition at line 390 of file [main.c](#).

4.19.2.12 copy_trim()

```
char * copy_trim (
    char const * s )
```

Definition at line 851 of file [main.c](#).

4.19.2.13 emb_optOut()

```
char * emb_optOut (
    EmbReal num,
    char * str )
```

Optimizes the number (*num*) for output to a text file and returns it as a string (*str*).

Definition at line 868 of file [main.c](#).

4.19.2.14 emb_readline()

```
int emb_readline (
    FILE * file,
    char * line,
    int maxLength )
```

Definition at line 809 of file [main.c](#).

4.19.2.15 embArc_print()

```
void embArc_print (
    EmbArc arc )
```

Definition at line [80](#) of file [main.c](#).

4.19.2.16 embColor_distance()

```
int embColor_distance (
    EmbColor a,
    EmbColor b )
```

Definition at line [671](#) of file [main.c](#).

4.19.2.17 embColor_read()

```
void embColor_read (
    FILE * f,
    EmbColor * c,
    int toRead )
```

Definition at line [681](#) of file [main.c](#).

4.19.2.18 embColor_write()

```
void embColor_write (
    FILE * f,
    EmbColor c,
    int toWrite )
```

Definition at line [691](#) of file [main.c](#).

4.19.2.19 embSatinOutline_generateSatinOutline()

```
void embSatinOutline_generateSatinOutline (
    EmbArray * lines,
    EmbReal thickness,
    EmbSatinOutline * result )
```

Definition at line [520](#) of file [main.c](#).

4.19.2.20 embSatinOutline_renderStitches()

```
EmbArray * embSatinOutline_renderStitches (
    EmbSatinOutline * result,
    EmbReal density )
```

Definition at line 608 of file [main.c](#).

4.19.2.21 embThread_findNearestColor()

```
int embThread_findNearestColor (
    EmbColor color,
    EmbColor * color_list,
    int n_colors )
```

Returns the closest color to the required color based on a list of available threads. The algorithm is a simple least squares search against the list. If the (square of) Euclidean 3-dimensional distance between the points in (red, green, blue) space is smaller then the index is saved and the remaining index is returned to the caller.

Parameters

<i>color</i>	The EmbColor color to match.
<i>colors</i>	The EmbThreadList pointer to start the search at.
<i>mode</i>	Is the argument an array of threads (0) or colors (1)?

Returns

closestIndex The entry in the ThreadList that matches.

Definition at line 715 of file [main.c](#).

4.19.2.22 embThread_findNearestThread()

```
int embThread_findNearestThread (
    EmbColor color,
    EmbThread * thread_list,
    int n_threads )
```

Definition at line 731 of file [main.c](#).

4.19.2.23 `embThread_getRandom()`

```
EmbThread embThread_getRandom (
    void    )
```

Returns a random thread color, useful in filling in cases where the actual color of the thread doesn't matter but one needs to be declared to test or render a pattern.

Returns

c The resulting color.

Definition at line [754](#) of file [main.c](#).

4.19.2.24 `embTime_initNow()`

```
void embTime_initNow (
    EmbTime * t    )
```

Definition at line [885](#) of file [main.c](#).

4.19.2.25 `embTime_time()`

```
EmbTime embTime_time (
    EmbTime * t    )
```

Definition at line [905](#) of file [main.c](#).

4.19.2.26 `embVector_print()`

```
void embVector_print (
    EmbVector v,
    char * label )
```

Definition at line [74](#) of file [main.c](#).

4.19.2.27 `entriesInDifatSector()`

```
unsigned int entriesInDifatSector (
    bcf_file_difat * fat )
```

Definition at line [308](#) of file [main.c](#).

4.19.2.28 get_trim_bounds()

```
void get_trim_bounds (
    char const * s,
    char const ** firstWord,
    char const ** trailingSpace )
```

Definition at line [839](#) of file [main.c](#).

4.19.2.29 GetFile()

```
FILE * GetFile (
    bcf_file * bcfFile,
    FILE * file,
    char * fileToFind )
```

Definition at line [233](#) of file [main.c](#).

4.19.2.30 haveExtraDIFATSectors()

```
int haveExtraDIFATSectors (
    bcf_file * file )
```

Definition at line [172](#) of file [main.c](#).

4.19.2.31 loadFatFromSector()

```
void loadFatFromSector (
    bcf_file_fat * fat,
    FILE * file )
```

Definition at line [481](#) of file [main.c](#).

4.19.2.32 parseDIFATSectors()

```
void parseDIFATSectors (
    FILE * file,
    bcf_file * bcfFile )
```

Definition at line [185](#) of file [main.c](#).

4.19.2.33 parseDirectoryEntryName()

```
void parseDirectoryEntryName (
    FILE * file,
    bcf_directory_entry * dir )
```

Definition at line [347](#) of file [main.c](#).

4.19.2.34 parseTime()

```
EmbTime parseTime (
    FILE * file )
```

Definition at line [372](#) of file [main.c](#).

4.19.2.35 readFullSector()

```
unsigned int readFullSector (
    FILE * file,
    bcf_file_difat * bcfFile,
    unsigned int * difatEntriesToRead )
```

Definition at line [314](#) of file [main.c](#).

4.19.2.36 readNextSector()

```
void readNextSector (
    FILE * file,
    bcf_directory * dir )
```

Definition at line [426](#) of file [main.c](#).

4.19.2.37 sectorSize()

```
unsigned int sectorSize (
    bcf_file * bcfFile )
```

Definition at line [162](#) of file [main.c](#).

4.19.2.38 seekToSector()

```
int seekToSector (
    bcf_file * bcfFile,
    FILE * file,
    const unsigned int sector )
```

Definition at line [178](#) of file [main.c](#).

4.19.2.39 stringInArray()

```
int stringInArray (
    const char * s,
    const char ** array )
```

Tests for the presence of a string *s* in the supplied *array*.

The end of the array is marked by an empty string.

Returns

0 if not present 1 if present.

Definition at line [797](#) of file [main.c](#).

4.19.2.40 write_24bit()

```
void write_24bit (
    FILE * file,
    int x )
```

Definition at line [660](#) of file [main.c](#).

4.19.3 Variable Documentation

4.19.3.1 black_thread

```
EmbThread black_thread = { { 0, 0, 0 }, "Black", "Black" }
```

Definition at line [56](#) of file [main.c](#).

4.19.3.2 difatEntriesInHeader

```
const unsigned int difatEntriesInHeader = 109
```

Definition at line [64](#) of file [main.c](#).

4.19.3.3 emb_error

```
int emb_error = 0
```

Definition at line [58](#) of file [main.c](#).

4.19.3.4 emb_verbose

```
int emb_verbose = 0
```

Definition at line [57](#) of file [main.c](#).

4.19.3.5 embConstantPi

```
const EmbReal embConstantPi = 3.1415926535
```

Definition at line [60](#) of file [main.c](#).

4.19.3.6 sizeOfChainingEntryAtEndOfDifatSector

```
const unsigned int sizeOfChainingEntryAtEndOfDifatSector = 4
```

Definition at line [67](#) of file [main.c](#).

4.19.3.7 sizeOfDifatEntry

```
const unsigned int sizeOfDifatEntry = 4
```

Definition at line [66](#) of file [main.c](#).

4.19.3.8 sizeOfDirectoryEntry

```
const unsigned int sizeOfDirectoryEntry = 128
```

Definition at line 68 of file [main.c](#).

4.19.3.9 sizeOfFatEntry

```
const unsigned int sizeOfFatEntry = sizeof(unsigned int)
```

Definition at line 65 of file [main.c](#).

4.19.3.10 WHITESPACE

```
char const WHITESPACE[] = " \t\n\r"
```

Definition at line 835 of file [main.c](#).

4.20 main.c

[Go to the documentation of this file.](#)

```
00001 /*  
00002 * This file is part of libembroidery.  
00003 *  
00004 * Copyright 2018-2023 The Embroidermodder Team  
00005 * Licensed under the terms of the zlib license.  
00006 */  
00007  
00008 #include <stdio.h>  
00009 #include <stdlib.h>  
00010 #include <string.h>  
00011 #include <math.h>  
00012 #include <time.h>  
00013  
00014 #include "embroidery_internal.h"  
00015  
00016 /* same order as flag_list, to use in jump table */  
00017 #define FLAG_TO 0  
00018 #define FLAG_TO_SHORT 1  
00019 #define FLAG_HELP 2  
00020 #define FLAG_HELP_SHORT 3  
00021 #define FLAG_FORMATS 4  
00022 #define FLAG_FORMATS_SHORT 5  
00023 #define FLAG QUIET 6  
00024 #define FLAG QUIET_SHORT 7  
00025 #define FLAG_VERBOSE 8  
00026 #define FLAG_VERBOSE_SHORT 9  
00027 #define FLAG VERSION 10  
00028 #define FLAG VERSION_SHORT 11  
00029 #define FLAG CIRCLE 12  
00030 #define FLAG CIRCLE_SHORT 13  
00031 #define FLAG ELLIPSE 14  
00032 #define FLAG ELLIPSE_SHORT 15  
00033 #define FLAG LINE 16  
00034 #define FLAG LINE_SHORT 17  
00035 #define FLAG POLYGON 18  
00036 #define FLAG POLYGON_SHORT 19  
00037 #define FLAG POLYLINE 20  
00038 #define FLAG POLYLINE_SHORT 21  
00039 #define FLAG RENDER 22  
00040 #define FLAG RENDER_SHORT 23
```

```

00041 #define FLAG_SATIN 24
00042 #define FLAG_SATIN_SHORT 25
00043 #define FLAG_STITCH 26
00044 #define FLAG_STITCH_SHORT 27
00045 #define FLAG_TEST 28
00046 #define FLAG_FULL_TEST_SUITE 29
00047 #define FLAG_HILBERT_CURVE 30
00048 #define FLAG_SIERPINSKI_TRIANGLE 31
00049 #define FLAG_FILL 32
00050 #define FLAG_FILL_SHORT 33
00051 #define FLAG_SIMULATE 34
00052 #define FLAG_COMBINE 35
00053 #define FLAG_CROSS_STITCH 36
00054 #define NUM_FLAGS 37
00055
00056 EmbThread black_thread = { { 0, 0, 0 }, "Black", "Black" };
00057 int emb_verbose = 0;
00058 int emb_error = 0;
00059
00060 const EmbReal embConstantPi = 3.1415926535;
00061
00062 /* Constant representing the number of EmbReal Indirect FAT
00063 * entries in a single header */
00064 const unsigned int difatEntriesInHeader = 109;
00065 const unsigned int sizeOFFatEntry = sizeof(unsigned int);
00066 const unsigned int sizeOfdifatEntry = 4;
00067 const unsigned int sizeOfChainingEntryAtEndOfDifatSector = 4;
00068 const unsigned int sizeOfDirectoryEntry = 128;
00069 /*
00070 const int supportedMinorVersion = 0x003E;
00071 const int littleEndianByteOrderMark = 0xFFFE;
00072 */
00073
00074 void embVector_print(EmbVector v, char *label)
00075 {
00076     printf("%sX = %f\n", label, v.x);
00077     printf("%sY = %f\n", label, v.y);
00078 }
00079
00080 void embArc_print(EmbArc arc)
00081 {
00082     embVector_print(arc.start, "start");
00083     embVector_print(arc.mid, "middle");
00084     embVector_print(arc.end, "end");
00085 }
00086
00087 #ifdef LIBEMBROIDERY_CLI
00088 /* DATA
00089 *****/
00090
00091
00092 const char *flag_list[] = {
00093     "--to",
00094     "-t",
00095     "--help",
00096     "-h",
00097     "--formats",
00098     "-F",
00099     "--quiet",
00100     "-q",
00101     "--verbose",
00102     "-V",
00103     "--version",
00104     "-v",
00105     "--circle",
00106     "-c",
00107     "--ellipse",
00108     "-e",
00109     "--line",
00110     "-l",
00111     "--polyline",
00112     "-P",
00113     "--polygon",
00114     "-p",
00115     "--render",
00116     "-r",
00117     "--satin",
00118     "-s",
00119     "--stitch",
00120     "-S",
00121     "--test",
00122     "--full-test-suite",
00123     "--hilbert-curve",
00124     "--sierpinski-triangle",
00125     "--fill",
00126     "-f",
00127     "--simulate",

```

```

00128     "--combine",
00129     "--cross-stitch"
00130 };
00131
00132 const char *version_string = "embroider v0.1";
00133 const char *welcome_message = "EMBROIDER\n"
00134     "    A command line program for machine embroidery.\n"
00135     "    Copyright 2013-2022 The Embroidermodder Team.\n"
00136     "    Licensed under the terms of the zlib license.\n"
00137     "\n"
00138     "    https://github.com/Embroidermodder/libembroidery\n"
00139     "    https://embroidermodder.org\n";
00140 #endif
00141
00142
00143 /* Checks that there are enough bytes to interpret the header,
00144 * stops possible segfaults when reading in the header bytes.
00145 *
00146 * Returns 0 if there aren't enough, or the length of the file
00147 * if there are.
00148 */
00149 int check_header_present(FILE* file, int minimum_header_length)
00150 {
00151     int length;
00152     fseek(file, 0, SEEK_END);
00153     length = ftell(file);
00154     fseek(file, 0, SEEK_SET);
00155     if (length < minimum_header_length) {
00156         return 0;
00157     }
00158     return length;
00159 }
00160
00161 unsigned int
00162 sectorSize(bcf_file* bcfFile)
00163 {
00164     /* version 3 uses 512 byte */
00165     if (bcfFile->header.majorVersion == 3) {
00166         return 512;
00167     }
00168     return 4096;
00169 }
00170
00171 int
00172 haveExtraDIFATSectors(bcf_file* file)
00173 {
00174     return (int)(entriesInDifatSector(file->difat) > 0);
00175 }
00176
00177 int
00178 seekToSector(bcf_file* bcfFile, FILE* file, const unsigned int sector)
00179 {
00180     unsigned int offset = sector * sectorSize(bcfFile) + sectorSize(bcfFile);
00181     return fseek(file, offset, SEEK_SET);
00182 }
00183
00184 void
00185 parseDIFATSectors(FILE* file, bcf_file* bcfFile)
00186 {
00187     unsigned int difatEntriesToRead = bcfFile->header.numberOfFATSectors - difatEntriesInHeader;
00188     unsigned int difatSectorNumber = bcfFile->header.firstDifatSectorLocation;
00189     while ((difatSectorNumber != CompoundFileSector_EndOfChain) && (difatEntriesToRead > 0)) {
00190         seekToSector(bcfFile, file, difatSectorNumber);
00191         difatSectorNumber = readFullSector(file, bcfFile->difat, &difatEntriesToRead);
00192     }
00193 }
00194
00195 int
00196 bcfFile_read(FILE* file, bcf_file* bcfFile)
00197 {
00198     unsigned int i, numberOfDirectoryEntriesPerSector;
00199     unsigned int directorySectorToReadFrom;
00200
00201     bcfFile->header = bcfFileHeader_read(file);
00202     if (memcmp(bcfFile->header.signature, "\xd0\xcf\x11\xe0\x1a\xb1\x1a\xe1", 8) != 0) {
00203         printf("bad header signature\n");
00204         printf("Failed to parse header\n");
00205         return 0;
00206     }
00207
00208     bcfFile->difat = bcf_difat_create(file, bcfFile->header.numberOfFATSectors, sectorSize(bcfFile));
00209     if (haveExtraDIFATSectors(bcfFile)) {
00210         parseDIFATSectors(file, bcfFile);
00211     }
00212
00213     bcfFile->fat = bcfFileFat_create(sectorSize(bcfFile));
00214     for (i = 0; i < bcfFile->header.numberOfFATSectors; ++i) {

```

```

00215     unsigned int fatSectorNumber = bcfFile->difat->fatSectorEntries[i];
00216     seekToSector(bcffile, file, fatSectorNumber);
00217     loadFatFromSector(bcfFile->fat, file);
00218 }
00219
00220     numberOfDirectoryEntriesPerSector = sectorSize(bcffile) / sizeOfDirectoryEntry;
00221     bcfFile->directory = CompoundFileDirectory(numberOfDirectoryEntriesPerSector);
00222     directorySectorToReadFrom = bcfFile->header.firstDirectorySectorLocation;
00223     while (directorySectorToReadFrom != CompoundFileSector_EndOfChain) {
00224         seekToSector(bcffile, file, directorySectorToReadFrom);
00225         readNextSector(file, bcfFile->directory);
00226         directorySectorToReadFrom = bcfFile->fat->fatEntries[directorySectorToReadFrom];
00227     }
00228
00229     return 1;
00230 }
00231
00232 FILE*
00233 GetFile(bcf_file* bcfFile, FILE* file, char* fileToFind)
00234 {
00235     int filesize, sectorSize, currentSector;
00236     int sizeToWrite, currentSize, totalSectors, i, j;
00237     FILE* fileOut = tmpfile();
00238     bcf_directory_entry* pointer = bcfFile->directory->dirEntries;
00239     while (pointer) {
00240         if (strcmp(fileToFind, pointer->directoryEntryName) == 0)
00241             break;
00242         pointer = pointer->next;
00243     }
00244     filesize = pointer->streamSize;
00245     sectorSize = bcfFile->difat->sectorSize;
00246     currentSize = 0;
00247     currentSector = pointer->startingSectorLocation;
00248     totalSectors = (int)ceil((float)filesize / sectorSize);
00249     for (i = 0; i < totalSectors; i++) {
00250         seekToSector(bcfFile, file, currentSector);
00251         sizeToWrite = filesize - currentSize;
00252         if (sectorSize < sizeToWrite) {
00253             sizeToWrite = sectorSize;
00254         }
00255         for (j=0; j<sizeToWrite; j++) {
00256             char input;
00257             fread(&input, 1, 1, file);
00258             fwrite(&input, 1, 1, fileOut);
00259         }
00260         currentSize += sizeToWrite;
00261         currentSector = bcfFile->fat->fatEntries[currentSector];
00262     }
00263     return fileOut;
00264 }
00265
00266 void
00267 bcf_file_free(bcf_file* bcfFile)
00268 {
00269     safe_free(bcfFile->difat);
00270     safe_free(bcfFile->fat);
00271     bcf_directory_free(&bcfFile->directory);
00272     free(bcfFile);
00273 }
00274
00275 bcf_file_difat*
00276 bcf_difat_create(FILE* file, unsigned int fatSectors, const unsigned int sectorSize)
00277 {
00278     unsigned int i;
00279     bcf_file_difat* difat = 0;
00280     unsigned int sectorRef;
00281
00282     difat = (bcf_file_difat*)malloc(sizeof(bcf_file_difat));
00283     if (!difat) {
00284         printf("ERROR: compound-file-difat.c bcf_difat_create(), cannot allocate memory for difat\n");
00285         return NULL;
00286     }
00287
00288     difat->sectorSize = sectorSize;
00289     if (fatSectors > difatEntriesInHeader) {
00290         fatSectors = difatEntriesInHeader;
00291     }
00292
00293     for (i = 0; i < fatSectors; ++i) {
00294         embInt_read(file, "sectorRef", &sectorRef, EMB_INT32_LITTLE);
00295         difat->fatSectorEntries[i] = sectorRef;
00296     }
00297     difat->fatSectorCount = fatSectors;
00298     for (i = fatSectors; i < difatEntriesInHeader; ++i) {
00299         embInt_read(file, "sectorRef", &sectorRef, EMB_INT32_LITTLE);
00300         if (sectorRef != CompoundFileSector_FreeSector) {
00301             printf("ERROR: compound-file-difat.c bcf_difat_create(), Unexpected sector value %x at "

```

```

        DIFAT[%d]\n", sectorRef, i);
00302     }
00303 }
00304     return difat;
00305 }
00306
00307 unsigned int
00308 entriesInDifatSector(bcf_file_difat* fat)
00309 {
00310     return (fat->sectorSize - sizeOfChainingEntryAtEndOfDifatSector) / sizeOfDifatEntry;
00311 }
00312
00313 unsigned int
00314 readFullSector(FILE* file, bcf_file_difat* bcfFile,
00315                         unsigned int* difatEntriesToRead)
00316 {
00317     unsigned int i;
00318     unsigned int sectorRef;
00319     unsigned int nextDifatSectorInChain;
00320     unsigned int entriesToReadInThisSector = 0;
00321     if (*difatEntriesToRead > entriesInDifatSector(bcfFile)) {
00322         entriesToReadInThisSector = entriesInDifatSector(bcfFile);
00323         *difatEntriesToRead -= entriesToReadInThisSector;
00324     }
00325     else {
00326         entriesToReadInThisSector = *difatEntriesToRead;
00327         *difatEntriesToRead = 0;
00328     }
00329
00330     for (i = 0; i < entriesToReadInThisSector; ++i) {
00331         embInt_read(file, "sectorRef", &sectorRef, EMB_INT32_LITTLE);
00332         bcfFile->fatSectorEntries[bcfFile->fatSectorCount] = sectorRef;
00333         bcfFile->fatSectorCount++;
00334     }
00335     for (i = entriesToReadInThisSector; i < entriesInDifatSector(bcfFile); ++i) {
00336         embInt_read(file, "sectorRef", &sectorRef, EMB_INT32_LITTLE);
00337         if (sectorRef != CompoundFileSector_FreeSector) {
00338             printf("ERROR: compound-file-difat.c readFullSector(), ");
00339             printf("Unexpected sector value %x at DIFAT[%d]\n", sectorRef, i);
00340         }
00341     }
00342     embInt_read(file, "nextDifatSectorInChain", &nextDifatSectorInChain, EMB_INT32_LITTLE);
00343     return nextDifatSectorInChain;
00344 }
00345
00346 void
00347 parseDirectoryEntryName(FILE* file, bcf_directory_entry* dir)
00348 {
00349     int i;
00350     for (i = 0; i < 32; ++i) {
00351         unsigned short unicodechar;
00352         embInt_read(file, "unicode char", &unicodechar, EMB_INT16_LITTLE);
00353         if (unicodechar != 0x0000) {
00354             dir->directoryEntryName[i] = (char)unicodechar;
00355         }
00356     }
00357 }
00358
00359 bcf_directory*
00360 CompoundFileDialogue(const unsigned int maxNumberOfDirectoryEntries)
00361 {
00362     bcf_directory* dir = (bcf_directory*)malloc(sizeof(bcf_directory));
00363     if (!dir) {
00364         printf("ERROR: compound-file-directory.c CompoundFileDialogue(), cannot allocate memory for
dir\n");
00365     } /* TODO: avoid crashing. null pointer will be accessed */
00366     dir->maxNumberOfDirectoryEntries = maxNumberOfDirectoryEntries;
00367     dir->dirEntries = 0;
00368     return dir;
00369 }
00370
00371 EmbTime
00372 parseTime(FILE* file)
00373 {
00374     EmbTime returnVal;
00375     unsigned int ft_low, ft_high;
00376     /*embTime_time(&returnVal); TODO: use embTime_time() rather than time(). */
00377     embInt_read(file, "ft_low", &ft_low, EMB_INT32_LITTLE);
00378     embInt_read(file, "ft_high", &ft_high, EMB_INT32_LITTLE);
00379     /* TODO: translate to actual date time */
00380     returnVal.day = 1;
00381     returnVal.hour = 2;
00382     returnVal.minute = 3;
00383     returnVal.month = 4;
00384     returnVal.second = 5;
00385     returnVal.year = 6;
00386     return returnVal;

```

```

00387 }
00388
00389 bcf_directory_entry*
00390 CompoundFileDirectoryEntry(FILE* file)
00391 {
00392     const int guidSize = 16;
00393     bcf_directory_entry* dir = malloc(sizeof(bcf_directory_entry));
00394     if (dir == NULL) {
00395         printf("ERROR: compound-file-directory.c CompoundFileDirectoryEntry(), cannot allocate memory
for dir\n");
00396         return 0;
00397     }
00398     memset(dir->directoryEntryName, 0, 32);
00399     parseDirectoryName(file, dir);
00400     dir->next = 0;
00401     dir->directoryEntryNameLength = fread_uint16(file);
00402     dir->objectType = (unsigned char)fgetc(file);
00403     if ((dir->objectType != ObjectTypeStorage) && (dir->objectType != ObjectTypeStream) &&
(dir->objectType != ObjectTypeRootEntry)) {
00404         printf("ERROR: compound-file-directory.c CompoundFileDirectoryEntry()");
00405         printf(", unexpected object type: %d\n", dir->objectType);
00406         return 0;
00407     }
00408     dir->colorFlag = (unsigned char)fgetc(file);
00409     embInt_read(file, "left sibling id", &(dir->leftSiblingId), EMB_INT32_LITTLE);
00410     embInt_read(file, "right sibling id", &(dir->rightSiblingId), EMB_INT32_LITTLE);
00411     embInt_read(file, "child id", &(dir->childId), EMB_INT32_LITTLE);
00412     fread(dir->CLSID, 1, guidSize, file);
00413     embInt_read(file, "state bits", &(dir->stateBits), EMB_INT32_LITTLE);
00414     dir->creationTime = parseTime(file);
00415     dir->modifiedTime = parseTime(file);
00416     embInt_read(file, "starting sector location", &(dir->startingSectorLocation), EMB_INT32_LITTLE);
00417     /* StreamSize should really be __int64 or long long,
00418      * but for our uses we should never run into an issue */
00419     embInt_read(file, "dir->streamSize", &(dir->streamSize), EMB_INT32_LITTLE);
00420     /* top portion of int64 */
00421     embInt_read(file, "dir->streamSizeHigh", &(dir->streamSizeHigh), EMB_INT32_LITTLE);
00422     return dir;
00423 }
00424
00425 void
00426 readNextSector(FILE* file, bcf_directory* dir)
00427 {
00428     unsigned int i;
00429     for (i = 0; i < dir->maxNumberOfDirectoryEntries; ++i) {
00430         bcf_directory_entry* dirEntry = CompoundFileDirectoryEntry(file);
00431         bcf_directory_entry* pointer = dir->dirEntries;
00432         if (!pointer) {
00433             dir->dirEntries = dirEntry;
00434         }
00435         else {
00436             while (pointer) {
00437                 if (!pointer->next) {
00438                     pointer->next = dirEntry;
00439                     break;
00440                 }
00441                 pointer = pointer->next;
00442             }
00443         }
00444     }
00445 }
00446
00447 void
00448 bcf_directory_free(bcf_directory** dir)
00449 {
00450     bcf_directory *dirptr;
00451     bcf_directory_entry* pointer;
00452     if (dir == NULL) {
00453         return;
00454     }
00455     dirptr = *dir;
00456     pointer = dirptr->dirEntries;
00457     while (pointer) {
00458         bcf_directory_entry* entryToFree;
00459         entryToFree = pointer;
00460         pointer = pointer->next;
00461         free(entryToFree);
00462     }
00463     safe_free(*dir);
00464 }
00465
00466 bcf_file_fat*
00467 bcfFileFat_create(const unsigned int sectorSize)
00468 {
00469     bcf_file_fat* fat = (bcf_file_fat*)malloc(sizeof(bcf_file_fat));
00470     if (!fat) {
00471         printf("ERROR: compound-file-fat.c bcfFileFat_create(), ");

```

```

00472     printf("cannot allocate memory for fat\n");
00473     return NULL;
00474 }
00475 fat->numberOfEntriesInFatSector = sectorSize / sizeOfFatEntry;
00476 fat->fatEntryCount = 0;
00477 return fat;
00478 }
00479
00480 void
00481 loadFatFromSector(bcf_file_fat* fat, FILE* file)
00482 {
00483     unsigned int i;
00484     unsigned int current_fat_entries = fat->fatEntryCount;
00485     unsigned int newSize = current_fat_entries + fat->numberOfEntriesInFatSector;
00486     for (i = current_fat_entries; i < newSize; ++i) {
00487         int fatEntry;
00488         embInt_read(file, "fatEntry", &fatEntry, EMB_INT32_LITTLE);
00489         fat->fatEntries[i] = fatEntry;
00490     }
00491     fat->fatEntryCount = newSize;
00492 }
00493
00494 bcf_file_header
00495 bcfFileHeader_read(FILE* file)
00496 {
00497     bcf_file_header header;
00498     fread(header.signature, 1, 8, file);
00499     fread(header.CLSID, 1, 16, file);
00500     header.minorVersion = fread_uint16(file);
00501     header.majorVersion = fread_uint16(file);
00502     header.byteOrder = fread_uint16(file);
00503     header.sectorShift = fread_uint16(file);
00504     header.miniSectorShift = fread_uint16(file);
00505     header.reserved1 = fread_uint16(file);
00506     embInt_read(file, "reserved2", &(header.reserved2), EMB_INT32_LITTLE);
00507     embInt_read(file, "numberOfDirectorySectors", &(header.numberOfDirectorySectors),
00508                 EMB_INT32_LITTLE);
00508     embInt_read(file, "numberOfFATSectors", &(header.numberOfFATSectors), EMB_INT32_LITTLE);
00509     embInt_read(file, "first dir sector location", &(header.firstDirectorySectorLocation),
00510                 EMB_INT32_LITTLE);
00510     embInt_read(file, "transaction signature number", &(header.transactionSignatureNumber),
00511                 EMB_INT32_LITTLE);
00511     embInt_read(file, "mini stream cutoff sector", &(header.miniStreamCutoffSize), EMB_INT32_LITTLE);
00512     embInt_read(file, "first mini fat sector", &(header.firstMiniFATSectorLocation),
00513                 EMB_INT32_LITTLE);
00513     embInt_read(file, "mini fat sectors", &(header.numberOfMiniFatSectors), EMB_INT32_LITTLE);
00514     embInt_read(file, "first difat Sector", &(header.firstDifatSectorLocation), EMB_INT32_LITTLE);
00515     embInt_read(file, "difat sectors", &(header.numberOfDifatSectors), EMB_INT32_LITTLE);
00516     return header;
00517 }
00518
00519 void
00520 embSatinOutline_generateSatinOutline(EmbArray *lines, EmbReal thickness, EmbSatinOutline* result)
00521 {
00522     int i;
00523     EmbLine line1, line2;
00524     EmbSatinOutline outline;
00525     EmbVector out;
00526     EmbVector v1;
00527     EmbVector temp;
00528     EmbLine line;
00529
00530     EmbReal halfThickness = thickness / 2.0;
00531     int intermediateOutlineCount = 2 * lines->count - 2;
00532     outline.side1 = embArray_create(EMB_VECTOR);
00533     if (!outline.side1) {
00534         printf("ERROR: emb-satin-line.c embSatinOutline_generateSatinOutline(), cannot allocate memory
for outline->side1\n");
00535         return;
00536     }
00537     outline.side2 = embArray_create(EMB_VECTOR);
00538     if (!outline.side2) {
00539         printf("ERROR: emb-satin-line.c embSatinOutline_generateSatinOutline(), cannot allocate memory
for outline->side2\n");
00540         return;
00541     }
00542
00543     for (i = 1; i < lines->count; i++) {
00544         line.start = lines->geometry[i - 1].object.vector;
00545         line.end = lines->geometry[i].object.vector;
00546
00547         embLine_normalVector(line, &v1, 1);
00548
00549         embVector_multiply(v1, halfThickness, &temp);
00550         temp = embVector_add(temp, lines->geometry[i-1].object.vector);
00551         embArray_addVector(outline.side1, temp);
00552         temp = embVector_add(temp, lines->geometry[i].object.vector);
00553
00554     }
00555 }
```

```

00553     embArray_addVector(outline.side1, temp);
00554
00555     embVector_multiply(v1, -halfThickness, &temp);
00556     temp = embVector_add(temp, lines->geometry[i - 1].object.vector);
00557     embArray_addVector(outline.side2, temp);
00558     temp = embVector_add(temp, lines->geometry[i].object.vector);
00559     embArray_addVector(outline.side2, temp);
00560 }
00561
00562 if (!result) {
00563     printf("ERROR: emb-satin-line.c embSatinOutline_generateSatinOutline(), result argument is
00564         null\n");
00565     return;
00566 }
00567 result->side1 = embArray_create(EMB_VECTOR);
00568 if (!result->side1) {
00569     printf("ERROR: emb-satin-line.c embSatinOutline_generateSatinOutline(), cannot allocate memory
00570         for result->side1\n");
00571     return;
00572 }
00573 result->side2 = embArray_create(EMB_VECTOR);
00574 if (!result->side2) {
00575     printf("ERROR: emb-satin-line.c embSatinOutline_generateSatinOutline(), cannot allocate memory
00576         for result->side2\n");
00577     return;
00578 }
00579 embArray_addVector(result->side1, outline.side1->geometry[0].object.vector);
00580 embArray_addVector(result->side2, outline.side2->geometry[0].object.vector);
00581
00582 for (i = 3; i < intermediateOutlineCount; i += 2) {
00583     line1.start = outline.side1->geometry[i - 3].object.vector;
00584     line1.end = outline.side1->geometry[i - 2].object.vector;
00585     line2.start = outline.side1->geometry[i - 1].object.vector;
00586     line2.end = outline.side1->geometry[i].object.vector;
00587     out = embLine_intersectionPoint(line1, line2);
00588     if (emb_error) {
00589         puts("No intersection point.");
00590     }
00591     embArray_addVector(result->side1, out);
00592
00593     line1.start = outline.side2->geometry[i - 3].object.vector;
00594     line1.end = outline.side2->geometry[i - 2].object.vector;
00595     line2.start = outline.side2->geometry[i - 1].object.vector;
00596     line2.end = outline.side2->geometry[i].object.vector;
00597     out = embLine_intersectionPoint(line1, line2);
00598     if (emb_error) {
00599         puts("No intersection point.");
00600     }
00601     embArray_addVector(result->side2, out);
00602
00603 embArray_addVector(result->side1, outline.side1->geometry[2 * lines->count - 3].object.vector);
00604 embArray_addVector(result->side2, outline.side2->geometry[2 * lines->count - 3].object.vector);
00605 result->length = lines->count;
00606 }
00607 EmbArray*
00608 embSatinOutline_renderStitches(EmbSatinOutline* result, EmbReal density)
00609 {
00610     int i, j;
00611     EmbVector currTop, currBottom, topDiff, bottomDiff, midDiff;
00612     EmbVector midLeft, midRight, topStep, bottomStep;
00613     EmbArray* stitches = 0;
00614     int numberofSteps;
00615     EmbReal midLength;
00616
00617     if (!result) {
00618         printf("ERROR: emb-satin-line.c embSatinOutline_renderStitches(), result argument is null\n");
00619         return 0;
00620     }
00621
00622     if (result->length > 0) {
00623         for (j = 0; j < result->length - 1; j++) {
00624             EmbGeometry *g10 = &(result->side1->geometry[j+0]);
00625             EmbGeometry *g11 = &(result->side1->geometry[j+1]);
00626             EmbGeometry *g20 = &(result->side2->geometry[j+0]);
00627             EmbGeometry *g21 = &(result->side2->geometry[j+1]);
00628             topDiff = embVector_subtract(g10->object.vector, g11->object.vector);
00629             bottomDiff = embVector_subtract(g21->object.vector, g20->object.vector);
00630
00631             midLeft = embVector_average(g10->object.vector, g20->object.vector);
00632             midRight = embVector_average(g11->object.vector, g21->object.vector);
00633
00634             midDiff = embVector_subtract(midLeft, midRight);
00635             midLength = embVector_length(midDiff);
00636
00637             if (midLength > 0) {
00638                 numberofSteps = (int)(midLength / density) + 1;
00639                 if (stitches) {
00640                     EmbArray_free(stitches);
00641                 }
00642                 stitches = embArray_create(EMB_VECTOR);
00643                 if (!stitches) {
00644                     printf("ERROR: emb-satin-line.c embSatinOutline_renderStitches(), cannot allocate memory
00645                         for stitches\n");
00646                     return 0;
00647                 }
00648                 for (i = 1; i < numberofSteps; i++) {
00649                     EmbVector step = embVector_lerp(midLeft, midRight, i / (float)numberofSteps);
00650                     embArray_addVector(stitches, step);
00651                 }
00652             }
00653         }
00654     }
00655     return stitches;
00656 }

```

```
00637     numberOfSteps = (int) (midLength * density / 200);
00638     embVector_multiply(topDiff, 1.0/numberOfSteps, &topStep);
00639     embVector_multiply(bottomDiff, 1.0/numberOfSteps, &bottomStep);
00640     currTop = g10->object.vector;
00641     currBottom = g20->object.vector;
00642
00643     for (i = 0; i < numberOfSteps; i++) {
00644         if (!stitches) {
00645             stitches = embArray_create(EMB_VECTOR);
00646         }
00647         embArray_addVector(stitches, currTop);
00648         embArray_addVector(stitches, currBottom);
00649         currTop = embVector_add(currTop, topStep);
00650         currBottom = embVector_add(currBottom, bottomStep);
00651     }
00652 }
00653     embArray_addVector(stitches, currTop);
00654     embArray_addVector(stitches, currBottom);
00655 }
00656 return stitches;
00657 }
00658
00659 void
00660 write_24bit(FILE* file, int x)
00661 {
00662     unsigned char a[4];
00663     a[0] = (unsigned char)0;
00664     a[1] = (unsigned char)(x & 0xFF);
00665     a[2] = (unsigned char)((x >> 8) & 0xFF);
00666     a[3] = (unsigned char)((x >> 16) & 0xFF);
00667     fwrite(a, 1, 4, file);
00668 }
00669
00670 int
00671 embColor_distance(EmbColor a, EmbColor b)
00672 {
00673     int t;
00674     t = (a.r-b.r)*(a.r-b.r);
00675     t += (a.g-b.g)*(a.g-b.g);
00676     t += (a.b-b.b)*(a.b-b.b);
00677     return t;
00678 }
00679
00680 void
00681 embColor_read(FILE *f, EmbColor *c, int toRead)
00682 {
00683     unsigned char b[4];
00684     fread(b, 1, toRead, f);
00685     c->r = b[0];
00686     c->g = b[1];
00687     c->b = b[2];
00688 }
00689
00690 void
00691 embColor_write(FILE *f, EmbColor c, int toWrite)
00692 {
00693     unsigned char b[4];
00694     b[0] = c.r;
00695     b[1] = c.g;
00696     b[2] = c.b;
00697     b[3] = 0;
00698     fwrite(b, 1, toWrite, f);
00699 }
00700
00714 int
00715 embThread_findNearestColor(EmbColor color, EmbColor *color_list, int n_colors)
00716 {
00717     int currentClosestValue = 256*256*3;
00718     int closestIndex = -1, i;
00719     for (i = 0; i < n_colors; i++) {
00720         int delta = embColor_distance(color, color_list[i]);
00721
00722         if (delta <= currentClosestValue) {
00723             currentClosestValue = delta;
00724             closestIndex = i;
00725         }
00726     }
00727     return closestIndex;
00728 }
00729
00730 int
00731 embThread_findNearestThread(EmbColor color, EmbThread *thread_list, int n_threads)
00732 {
00733     int currentClosestValue = 256*256*3;
00734     int closestIndex = -1, i;
00735     for (i = 0; i < n_threads; i++) {
00736         int delta = embColor_distance(color, thread_list[i].color);
```

```

00737
00738     if (delta <= currentClosestValue) {
00739         currentClosestValue = delta;
00740         closestIndex = i;
00741     }
00742 }
00743 return closestIndex;
00744 }
00745
00753 EmbThread
00754 embThread_getRandom(void)
00755 {
00756     EmbThread c;
00757     c.color.r = rand()%256;
00758     c.color.g = rand()%256;
00759     c.color.b = rand()%256;
00760     strcpy(c.description, "random");
00761     strcpy(c.catalogNumber, "");
00762     return c;
00763 }
00764
00765 void
00766 binaryReadString(FILE* file, char* buffer, int maxLength)
00767 {
00768     int i = 0;
00769     while(i < maxLength) {
00770         buffer[i] = (char)fgetc(file);
00771         if (buffer[i] == '\0') break;
00772         i++;
00773     }
00774 }
00775
00776 void
00777 binaryReadUnicodeString(FILE* file, char *buffer, const int stringLength)
00778 {
00779     int i = 0;
00800     for (i = 0; i < stringLength * 2; i++) {
00801         char input = (char)fgetc(file);
00802         if (input != 0) {
00803             buffer[i] = input;
00804         }
00805     }
00806 }
00807
00808 int
00809 emb_readline(FILE* file, char *line, int maxLength)
00810 {
00811     int i;
00812     char c;
00813     for (i = 0; i < maxLength-1; i++) {
00814         if (!fread(&c, 1, 1, file)) {
00815             break;
00816         }
00817         if (c == '\r') {
00818             fread(&c, 1, 1, file);
00819             if (c != '\n') {
00820                 fseek(file, -1L, SEEK_CUR);
00821             }
00822             break;
00823         }
00824         if (c == '\n') {
00825             break;
00826         }
00827         *line = c;
00828         line++;
00829     }
00830     *line = 0;
00831     return i;
00832 }
00833
00834 /* TODO: trimming function should handle any character, not just whitespace */
00835 char const WHITESPACE[] = "\t\n\r";
00836
00837 /* TODO: description */
00838 void

```

```

00839 get_trim_bounds(char const *s, char const **firstWord, char const **trailingSpace)
00840 {
00841     char const* lastWord = 0;
00842     *firstWord = lastWord = s + strspn(s, WHITESPACE);
00843     do {
00844         *trailingSpace = lastWord + strcspn(lastWord, WHITESPACE);
00845         lastWord = *trailingSpace + strspn(*trailingSpace, WHITESPACE);
00846     } while (*lastWord != '\0');
00847 }
00848
00849 /* TODO: description */
00850 char*
00851 copy_trim(char const *s)
00852 {
00853     char const *firstWord = 0, *trailingSpace = 0;
00854     char* result = 0;
00855     size_t newLength;
00856
00857     get_trim_bounds(s, &firstWord, &trailingSpace);
00858     newLength = trailingSpace - firstWord;
00859
00860     result = (char*)malloc(newLength + 1);
00861     memcpy(result, firstWord, newLength);
00862     result[newLength] = '\0';
00863     return result;
00864 }
00865
00866 char*
00867 emb_optOut(EmbReal num, char* str)
00868 {
00869     char *str_end;
00870     /* Convert the number to a string */
00871     sprintf(str, "%10f", num);
00872     /* Remove trailing zeroes */
00873     str_end = str + strlen(str);
00874     while (--str_end == '0');
00875     str_end[1] = 0;
00876     /* Remove the decimal point if it happens to be an integer */
00877     if (*str_end == '.') {
00878         *str_end = 0;
00879     }
00880     return str;
00881 }
00882 }
00883
00884 void
00885 embTime_initNow(EmbTime* t)
00886 {
00887 #ifdef ARDUINO
00888 /*TODO: arduino embTime_initNow */
00889 #else
00890     time_t rawtime;
00891     struct tm* timeinfo;
00892     time(&rawtime);
00893     timeinfo = localtime(&rawtime);
00894
00895     t->year    = timeinfo->tm_year;
00896     t->month   = timeinfo->tm_mon;
00897     t->day     = timeinfo->tm_mday;
00898     t->hour    = timeinfo->tm_hour;
00899     t->minute  = timeinfo->tm_min;
00900     t->second  = timeinfo->tm_sec;
00901 #endif /* ARDUINO */
00902 }
00903
00904 EmbTime
00905 embTime_time(EmbTime* t)
00906 {
00907 #ifdef ARDUINO
00908 /*TODO: arduino embTime_time */
00909 #else
00910
00911     int divideByZero = 0;
00912     divideByZero = divideByZero/divideByZero; /*TODO: wrap time() from time.h and verify it works
00913     consistently */
00914 #endif /* ARDUINO */
00915     return *t;
00916 }
00917
00918 #ifdef LIBEMBROIDERY_CLI
00919 void
00920 report(int result, char *label)
00921 {
00922     printf("%s Test...%*c", label, (int)(20-strlen(label)), ' ');
00923     if (result) {
00924         printf(RED_TERM_COLOR "[FAIL] [CODE=%d]\n" RESET_TERM_COLOR, result);
00925     }

```

```

00926     else {
00927         printf(GREEN_TERM_COLOR " [PASS]\n" RESET_TERM_COLOR);
00928     }
00929 }
00930
00931 void
00932 testMain(int level)
00933 {
00934     EmbPattern *pattern = embPattern_create();
00935     int overall = 0;
00936     int circleResult = testEmbCircle();
00937     int threadResult = testThreadColor();
00938     int formatResult = testEmbFormat();
00939     int arcResult = testGeomArc();
00940     int create1Result = create_test_file_1("test01.csv");
00941     int create2Result = create_test_file_2("test02.csv");
00942     int create3Result = create_test_file_3("test03.csv");
00943     int svg1Result = convert("test01.csv", "test01.svg");
00944     int svg2Result = convert("test02.csv", "test02.svg");
00945     int svg3Result = convert("test03.csv", "test03.svg");
00946     int dst1Result = convert("test01.csv", "test01.dst");
00947     int dst2Result = convert("test02.csv", "test02.dst");
00948     int dst3Result = convert("test03.csv", "test03.dst");
00949     int hilbertCurveResult = hilbert_curve(pattern, 3);
00950     int renderResult = embPattern_render(pattern, "hilbert_level_3.png");
00951     int simulateResult = embPattern_simulate(pattern, "hilbert_level_3.avi");
00952
00953     overall += circleResult;
00954     overall += threadResult;
00955     overall += formatResult;
00956     overall += arcResult;
00957     overall += create1Result;
00958     overall += create2Result;
00959     overall += create3Result;
00960     overall += svg1Result;
00961     overall += svg2Result;
00962     overall += svg3Result;
00963     overall += dst1Result;
00964     overall += dst2Result;
00965     overall += dst3Result;
00966
00967     if (emb_verbose >= 0) {
00968         puts("SUMMARY OF RESULTS");
00969         puts("-----");
00970         report(circleResult, "Tangent Point");
00971         report(threadResult, "Thread");
00972         report(formatResult, "Format");
00973         report(arcResult, "Arc");
00974         report(create1Result, "Create CSV 1");
00975         report(create2Result, "Create CSV 2");
00976         report(create3Result, "Create CSV 3");
00977         report(svg1Result, "Convert CSV-SVG 1");
00978         report(svg2Result, "Convert CSV-SVG 2");
00979         report(svg3Result, "Convert CSV-SVG 3");
00980         report(dst1Result, "Convert CSV-DST 1");
00981         report(dst2Result, "Convert CSV-DST 2");
00982         report(dst3Result, "Convert CSV-DST 3");
00983         report(hilbertCurveResult, "Generating Hilbert Curve");
00984         report(renderResult, "Rendering Hilbert Curve");
00985         report(simulateResult, "Simulating Hilbert Curve");
00986     }
00987
00988     embPattern_free(pattern);
00989     if (level > 0) {
00990         puts("More expensive tests.");
00991         full_test_matrix("test_matrix.txt");
00992     }
00993     if (overall == 0) {
00994         puts("PASS");
00995     }
00996     else {
00997         puts("FAIL");
00998     }
00999 }
01000
01001 void
01002 testTangentPoints(EmbCircle c, EmbVector p, EmbVector *t0, EmbVector *t1)
01003 {
01004     emb_error = 0;
01005     t0->x = 0.0;
01006     t0->y = 0.0;
01007     t1->x = 0.0;
01008     t1->y = 0.0;
01009     if (!getCircleTangentPoints(c, p, t0, t1)) {
01010         printf("Error calculating tangent points.\n");
01011         emb_error = 1;
01012     }

```

```

01013     else {
01014         if (emb_verbose > 0) {
01015             printf("Circle : cr=%f, cx=%f, cy=%f\n"
01016                 "Point : px=%f, py=%f\n"
01017                 "Tangent: tx0=%f, ty0=%f\n"
01018                 "Tangent: tx1=%f, ty1=%f\n",
01019                 c.radius, c.center.x, c.center.y,
01020                 p.x, p.y,
01021                 t0->x, t0->y,
01022                 t1->x, t1->y);
01023     }
01024 }
01025 }
01026
01027 int
01028 testEmbCircle(void)
01029 {
01030     EmbReal error;
01031     EmbReal epsilon = 1e-3;
01032     EmbVector p0, p1;
01033     /* Problem */
01034     EmbCircle c1 = {{0.0, 0.0}, 3.0};
01035     /* Solution */
01036     EmbVector t0 = {2.2500, 1.9843};
01037     EmbVector t1 = {2.2500, -1.9843};
01038     EmbVector p = {4.0, 0.0};
01039     /* Test */
01040     testTangentPoints(c1, p, &p0, &p1);
01041     error = embVector_distance(p0, t0) + embVector_distance(p1, t1);
01042     if (error > epsilon) {
01043         printf("Error larger than tolerance, circle test 1: %f.\n\n", error);
01044         return 16;
01045     }
01046
01047     return 0;
01048 }
01049
01050 int testEmbCircle_2(void) {
01051     EmbReal error;
01052     EmbReal epsilon = 1e-3;
01053     EmbVector p0, p1;
01054     EmbCircle c2 = {{20.1762, 10.7170}, 6.8221};
01055     /* Solution */
01056     EmbVector s0 = {19.0911, 17.4522};
01057     EmbVector s1 = {26.4428, 13.4133};
01058     EmbVector p = {24.3411, 18.2980};
01059     /* Test */
01060     testTangentPoints(c2, p, &p0, &p1);
01061     error = embVector_distance(p0, s0) + embVector_distance(p1, s1);
01062     if (error > epsilon) {
01063         printf("Error larger than tolerance, circle test 2: %f.\n\n", error);
01064         return 17;
01065     }
01066
01067     return 0;
01068 }
01069
01070 void
01071 printArcResults(EmbReal bulge, EmbArc arc,
01072                     EmbReal centerX,   EmbReal centerY,
01073                     EmbReal radius,    EmbReal diameter,
01074                     EmbReal chord,     EmbReal chordMidX,
01075                     EmbReal chordMidY, EmbReal sagitta,
01076                     EmbReal apothem,   EmbReal incAngle,
01077                     char   clockwise)
01078 {
01079     fprintf(stdout, "bulge      = %f\n"
01080             "startX     = %f\n"
01081             "startY     = %f\n"
01082             "endX       = %f\n"
01083             "endY       = %f\n"
01084             "midX       = %f\n"
01085             "midY       = %f\n"
01086             "centerX    = %f\n"
01087             "centerY    = %f\n"
01088             "radius      = %f\n"
01089             "diameter    = %f\n"
01090             "chord       = %f\n"
01091             "chordMidX  = %f\n"
01092             "chordMidY  = %f\n"
01093             "sagitta    = %f\n"
01094             "apothem    = %f\n"
01095             "incAngle   = %f\n"
01096             "clockwise  = %d\n"
01097             "\n",
01098             bulge,
01099             arc.start.x,

```

```

01100         arc.start.y,
01101         arc.end.x,
01102         arc.end.y,
01103         arc.mid.x,
01104         arc.mid.y,
01105         centerX,
01106         centerY,
01107         radius,
01108         diameter,
01109         chord,
01110         chordMidX,
01111         chordMidY,
01112         sagitta,
01113         apothem,
01114         incAngle,
01115         clockwise);
01116 }
01117
01118 int
01119 testGeomArc(void)
01120 {
01121     EmbArc arc;
01122     EmbVector center, chordMid;
01123     EmbReal bulge, radius, diameter, chord, sagitta, apothem, incAngle;
01124     char clockwise;
01125
01126     bulge = -0.414213562373095;
01127     arc.start.x = 1.0;
01128     arc.start.y = 0.0;
01129     arc.end.x = 2.0;
01130     arc.end.y = 1.0;
01131     if (getArcDataFromBulge(bulge, &arc,
01132             &(center.x), &(center.y),
01133             &radius, &diameter,
01134             &chord,
01135             &(chordMid.x), &(chordMid.y),
01136             &sagitta, &apothem,
01137             &incAngle, &clockwise)) {
01138         if (emb_verbose > 0) {
01139             fprintf(stdout, "Clockwise Test:\n");
01140             printArcResults(bulge, arc, center.x, center.y,
01141                             radius, diameter,
01142                             chord,
01143                             chordMid.x, chordMid.y,
01144                             sagitta, apothem,
01145                             incAngle, clockwise);
01146     }
01147 }
01148
01149     bulge = 2.414213562373095;
01150     arc.start.x = 4.0;
01151     arc.start.y = 0.0;
01152     arc.end.x = 5.0;
01153     arc.end.y = 1.0;
01154     if (getArcDataFromBulge(bulge, &arc,
01155             &(center.x), &(center.y),
01156             &radius, &diameter,
01157             &chord,
01158             &(chordMid.x), &(chordMid.y),
01159             &sagitta, &apothem,
01160             &incAngle, &clockwise)) {
01161         if (emb_verbose > 0) {
01162             fprintf(stdout, "Counter-Clockwise Test:\n");
01163             printArcResults(bulge, arc, center.x, center.y,
01164                             radius, diameter,
01165                             chord,
01166                             chordMid.x, chordMid.y,
01167                             sagitta, apothem,
01168                             incAngle, clockwise);
01169     }
01170 }
01171
01172     return 0;
01173 }
01174
01175 int
01176 testThreadColor(void)
01177 {
01178     unsigned int tColor = 0xFFD25F00;
01179     int tBrand = Sulky_Rayon;
01180     int tNum = threadColorNum(tColor, tBrand);
01181     const char* tName = threadColorName(tColor, tBrand);
01182
01183     if (emb_verbose > 0) {
01184         printf("Color : 0x%X\n"
01185               "Brand : %d\n"
01186               "Num   : %d\n"

```

```
01187         "Name    : %s\n\n",
01188         tColor,
01189         tBrand,
01190         tNum, /* Solution: 1833 */
01191         tName); /* Solution: Pumpkin Pie */
01192     }
01193     return 0;
01194 }
01195
01196 int testEmbFormat(void) {
01197     const char* tName = "example.zsk";
01198     int format = emb_identify_format(tName);
01199
01200     if (emb_verbose > 0) {
01201         printf("Filename   : %s\n"
01202             "Extension  : %s\n"
01203             "Description: %s\n"
01204             "Reader     : %c\n"
01205             "Writer     : %c\n"
01206             "Type       : %d\n\n",
01207             tName,
01208             formatTable[format].extension,
01209             formatTable[format].description,
01210             formatTable[format].reader_state,
01211             formatTable[format].writer_state,
01212             formatTable[format].type);
01213     }
01214
01215     if (strcmp(formatTable[format].extension, ".zsk")) return 20;
01216     if (strcmp(formatTable[format].description, "ZSK USA Embroidery Format")) {
01217         return 21;
01218     }
01219     if (formatTable[format].reader_state != 'U') return 22;
01220     if (formatTable[format].writer_state != ' ') return 23;
01221     if (formatTable[format].type != 1) return 24;
01222     return 0;
01223 }
01224
01225 int
01226 create_test_file_1(const char* outf)
01227 {
01228     int i;
01229     EmbPattern* p;
01230     EmbStitch st;
01231
01232     p = embPattern_create();
01233     if (!p) {
01234         puts("ERROR: convert(), cannot allocate memory for p\n");
01235         return 1;
01236     }
01237
01238     /* 10mm circle */
01239     for (i = 0; i < 20; i++) {
01240         embPattern_addStitchRel(p, 0.0, 1.0, JUMP, 0);
01241     }
01242     for (i = 0; i < 200; i++) {
01243         st.x = 10 + 10 * sin(i * (0.03141592));
01244         st.y = 10 + 10 * cos(i * (0.03141592));
01245         st.flags = NORMAL;
01246         st.color = 0;
01247         embPattern_addStitchAbs(p, st.x, st.y, st.flags, st.color);
01248     }
01249
01250     embPattern_addThread(p, black_thread);
01251     embPattern_end(p);
01252
01253     if (!embPattern_writeAuto(p, outf)) {
01254         return 16;
01255     }
01256
01257     embPattern_free(p);
01258     return 0;
01259 }
01260
01261 int
01262 create_test_file_2(const char* outf)
01263 {
01264     int i;
01265     EmbPattern* p;
01266     EmbStitch st;
01267
01268     p = embPattern_create();
01269     if (!p) {
01270         puts("ERROR: convert(), cannot allocate memory for p\n");
01271         return 1;
01272     }
01273 }
```

```

01274     /* sin wave */
01275     for (i = 0; i < 100; i++) {
01276         st.x = 10 + 10 * sin(i * (0.5 / 3.141592));
01277         st.y = 10 + i * 0.1;
01278         st.flags = NORMAL;
01279         st.color = 0;
01280         embPattern_addStitchAbs(p, st.x, st.y, st.flags, st.color);
01281     }
01282     embPattern_addThread(p, black_thread);
01283     embPattern_end(p);
01284
01285     if (!embPattern_writeAuto(p, outf)) {
01286         return 16;
01287     }
01288
01289     embPattern_free(p);
01290     return 0;
01291 }
01292
01293 int
01294 create_test_file_3(const char* outf)
01295 {
01296     EmbPattern* p;
01297     EmbCircle circle;
01298
01299     p = embPattern_create();
01300     if (!p) {
01301         puts("ERROR: convert(), cannot allocate memory for p\n");
01302         return 1;
01303     }
01304
01305     circle.center.x = 10.0;
01306     circle.center.y = 1.0;
01307     circle.radius = 5.0;
01308     embPattern_addCircleAbs(p, circle);
01309
01310     embPattern_addThread(p, black_thread);
01311     embPattern_convertGeometry(p);
01312     embPattern_end(p);
01313
01314     if (!embPattern_writeAuto(p, outf)) {
01315         return 16;
01316     }
01317 }
01318
01319     embPattern_free(p);
01320     return 0;
01321 }
01322
01323
01324 /* TODO: Add capability for converting multiple files of various types to a single format.
01325 Currently, we only convert a single file to multiple formats. */
01326 #if 0
01327 int testThreadColor(void) {
01328     unsigned int tColor = 0xFFD25F00;
01329     EmbColor c;
01330     c.r = 0xD2;
01331     c.g = 0x5F;
01332     c.b = 0x00;
01333     int tBrand = Sulky_Rayon;
01334     int tNum = threadColorNum(c, tBrand);
01335     char fName[50];
01336     threadColorName(fName, c, tBrand);
01337
01338     printf("Color : 0x%X\n"
01339             "Brand : %d\n"
01340             "Num   : %d\n"
01341             "Name  : %s\n\n",
01342             tColor,
01343             tBrand,
01344             tNum, /* Solution: 1833 */
01345             fName); /* Solution: Pumpkin Pie */
01346     return 0;
01347 }
01348#endif
01349
01350 /*
01351 * Table of from/to for formats. What conversions after a from A to B conversion
01352 * leave a file with the same render?
01353 *
01354 * Add command "--full-test-suite" for this full matrix.
01355 */
01356
01357 int
01358 full_test_matrix(char *fname)
01359 {
01360     int i, j, success, ntests;

```

```

01361     FILE *f;
01362     f = fopen(fname, "wb");
01363     if (!f) {
01364         puts("ERROR: full_test_matrix(fname) failed to open file.");
01365         return 1;
01366     }
01367
01368     success = 0;
01369     ntests = (numberOfFormats - 1)*(numberOfFormats - 5);
01370     for (i = 0; i < numberOfFormats; i++) {
01371         char fname[100];
01372         if (formatTable[i].color_only) {
01373             continue;
01374         }
01375         strcpy(fname, "test01");
01376         strcat(fname, formatTable[i].extension);
01377         create_test_file_1(fname);
01378         for (j=0; j < numberOfFormats; j++) {
01379             EmbPattern *pattern = 0;
01380             char fname_converted[100];
01381             char fname_image[100];
01382             int result;
01383             strcpy(fname_converted, "test01_");
01384             strcat(fname_converted, formatTable[i].extension+1);
01385             strcat(fname_converted, formatTable[j].extension);
01386             strcpy(fname_image, "test01_");
01387             strcat(fname_image, formatTable[i].extension+1);
01388             strcat(fname_image, "_");
01389             strcat(fname_image, formatTable[j].extension+1);
01390             strcat(fname_image, ".ppm");
01391             printf("Attempting: %s %s\n", fname, fname_converted);
01392             result = convert(fname, fname_converted);
01393             embPattern_readAuto(pattern, fname_converted);
01394             embPattern_render(pattern, fname_image);
01395             embPattern_free(pattern);
01396             fprintf(f, "%d %d %f% ", i, j, 100*success/(1.0*ntests));
01397             if (!result) {
01398                 fprintf(f, "PASS\n");
01399                 success++;
01400             } else {
01401                 fprintf(f, "FAIL\n");
01402             }
01403         }
01404     }
01405     fclose(f);
01406     return 0;
01407 }
01408 }
01409
01410 void
01411 usage(void)
01412 {
01413     puts(welcome_message);
01414     /* construct from tables above somehow, like how getopt_long works,
01415      * but we're avoiding that for compatibility
01416      * (C90, standard libraries only) */
01417     puts("Usage: embroider [OPTIONS] fileToRead... \n");
01418     puts("");
01419     puts("Conversion:");
01420     puts("    -t, --to           Convert all files given to the format specified");
01421     puts("                          by the arguments to the flag, for example:");
01422     puts("                                $ embroider -t dst input.pes");
01423     puts("                                would convert \"input.pes\" to \"input.dst\"");
01424     puts("                                in the same directory the program runs in.");
01425     puts("");
01426     puts("                         The accepted input formats are (TO BE DETERMINED).");
01427     puts("                         The accepted output formats are (TO BE DETERMINED).");
01428     puts("");
01429     puts("Output:");
01430     puts("    -h, --help          Print this message.");
01431     puts("    -F, --formats       Print help on the formats that embroider can deal with.");
01432     puts("    -q, --quiet          Only print fatal errors.");
01433     puts("    -V, --verbose         Print everything that has reporting.");
01434     puts("    -v, --version        Print the version.");
01435     puts("");
01436     puts("Modify patterns:");
01437     puts("    --combine           takes 3 arguments and combines the first");
01438     puts("                          two by placing them atop each other and");
01439     puts("                          outputs to the third");
01440     puts("                                $ embroider --combine a.dst b.dst output.dst");
01441     puts("");
01442     puts("Graphics:");
01443     puts("    -c, --circle         Add a circle defined by the arguments given to the current pattern.");
01444     puts("    -e, --ellipse        Add a circle defined by the arguments given to the current pattern.");
01445     puts("    -l, --line            Add a line defined by the arguments given to the current pattern.");
01446     puts("    -P, --polyline        Add a polyline.");
01447     puts("    -p, --polygon         Add a polygon.");

```

```

01448 puts(" -r, --render      Create an image in PNG format of what the embroidery should look
01449 like.");
01450 puts(" -s, --satin       Fill the current geometry with satin stitches according");
01451 puts(" -S, --stitch      Add a stitch defined by the arguments given to the current pattern.");
01452 puts("");
01453 puts("Quality Assurance:");
01454 puts(" --test          Run the basic test suite.");
01455 puts(" --full-test-suite Run all tests, even those we expect to fail.");
01456 }
01457
01458 void
01459 formats(void)
01460 {
01461     const char* extension = 0;
01462     const char* description = 0;
01463     char readerState;
01464     char writerState;
01465     int numReaders = 0;
01466     int numWriters = 0;
01467     int i;
01468     puts("List of Formats");
01469     puts("-----");
01470     puts("");
01471     puts("    KEY");
01472     puts("    'S' = Yes, and is considered stable.");
01473     puts("    'U' = Yes, but may be unstable.");
01474     puts("    ' ' = No.");
01475     puts("");
01476     printf(" Format      Read      Write      Description\n");
01477     printf("|_____|_____|_____|_____\n");
01478     printf("|_____|_____|_____|_____\n");
01479
01480     for (i = 0; i < numberFormats; i++) {
01481         extension = formatTable[i].extension;
01482         description = formatTable[i].description;
01483         readerState = formatTable[i].reader_state;
01484         writerState = formatTable[i].writer_state;
01485
01486         numReaders += readerState != ' ' ? 1 : 0;
01487         numWriters += writerState != ' ' ? 1 : 0;
01488         printf("| %4s | %c | %c | %-49s |\n", extension, readerState, writerState,
01489             description);
01490     }
01491     printf("|_____|_____|_____|_____\n");
01492     printf("|_____|_____|_____|_____\n");
01493     printf(" Total: | %3d | %3d |_____\n", numReaders, numWriters);
01494     printf("|_____|_____|_____|_____\n");
01495     puts("");
01496 }
01497
01498 void
01499 to_flag(char **argv, int argc, int i)
01500 {
01501     if (i + 2 < argc) {
01502         int j;
01503         char output_fname[100];
01504         int format;
01505         sprintf(output_fname, "example.%s", argv[i+1]);
01506         format = emb_identify_format(output_fname);
01507         if (format < 0) {
01508             puts("Error: format unrecognised.");
01509         }
01510         for (j=i+2; j<argc; j++) {
01511             int length = strlen(argv[j]);
01512             output_fname[0] = 0;
01513             strcpy(output_fname, argv[j]);
01514             output_fname[length-4] = 0;
01515             strcat(output_fname, formatTable[format].extension);
01516             printf("Converting %s to %s.\n",
01517                 argv[j], output_fname);
01518             convert(argv[j], output_fname);
01519         }
01520     }
01521     else {
01522         puts("Usage of the to flag is:");
01523         puts("   embroider -t FORMAT FILE(S)");
01524         puts("but it appears you entered less than 3 arguments to embroider.");
01525     }
01526 }
01527
01528 /* TODO: Add capability for converting multiple files of various types to a single format. Currently,
01529 we only convert a single file to multiple formats. */
01530 int
01531 command_line_interface(int argc, char* argv[])

```

```
01531 {
01532     EmbPattern *current_pattern = embPattern_create();
01533     int i, j, flags, result;
01534     if (argc == 1) {
01535         usage();
01536         return 0;
01537     }
01538     flags = argc-1;
01539     for (i=1; i < argc; i++) {
01540         result = -1;
01541         /* identify what flag index the user may have entered */
01542         for (j=0; j < NUM_FLAGS; j++) {
01543             if (!strcmp(flag_list[j], argv[i])) {
01544                 result = j;
01545                 break;
01546             }
01547         }
01548         /* apply the flag */
01549         switch (result) {
01550             case FLAG_TO:
01551             case FLAG_TO_SHORT:
01552                 to_flag(argv, argc, i);
01553                 break;
01554             case FLAG_HELP:
01555             case FLAG_HELP_SHORT:
01556                 usage();
01557                 break;
01558             case FLAG_FORMATS:
01559             case FLAG_FORMATS_SHORT:
01560                 formats();
01561                 break;
01562             case FLAG_QUIET:
01563             case FLAG_QUIET_SHORT:
01564                 emb_verbose = -1;
01565                 break;
01566             case FLAG_VERBOSE:
01567             case FLAG_VERBOSE_SHORT:
01568                 emb_verbose = 1;
01569                 break;
01570             case FLAG_CIRCLE:
01571             case FLAG_CIRCLE_SHORT:
01572                 puts("This flag is not implemented.");
01573                 break;
01574             case FLAG_ELLIPSE:
01575             case FLAG_ELLIPSE_SHORT:
01576                 puts("This flag is not implemented.");
01577                 break;
01578             case FLAG_LINE:
01579             case FLAG_LINE_SHORT:
01580                 puts("This flag is not implemented.");
01581                 break;
01582             case FLAG_POLYGON:
01583             case FLAG_POLYGON_SHORT:
01584                 puts("This flag is not implemented.");
01585                 break;
01586             case FLAG_POLYLINE:
01587             case FLAG_POLYLINE_SHORT:
01588                 puts("This flag is not implemented.");
01589                 break;
01590             case FLAG_SATIN:
01591             case FLAG_SATIN_SHORT:
01592                 puts("This flag is not implemented.");
01593                 break;
01594             case FLAG_STITCH:
01595             case FLAG_STITCH_SHORT:
01596                 puts("This flag is not implemented.");
01597                 break;
01598             case FLAG_SIERPINSKI_TRIANGLE:
01599                 puts("This flag is not implemented.");
01600                 break;
01601             case FLAG_FILL:
01602                 if (i + 3 < argc) {
01603                     EmbImage image;
01604                     /* the user appears to have entered the needed arguments */
01605                     i++;
01606                     /* to stb command */
01607                     image = embImage_create(2000, 2000);
01608                     embImage_read(&image, argv[i]);
01609                     i++;
01610                     embPattern_horizontal_fill(current_pattern, &image, atoi(argv[i]));
01611                     embImage_free(&image);
01612                     i++;
01613                     embPattern_writeAuto(current_pattern, argv[i]);
01614                 }
01615                 break;
01616             case FLAG_RENDER:
```

```

01618     case FLAG_RENDER_SHORT:
01619         if (i + 2 < argc) {
01620             /* the user appears to have entered filenames after render */
01621             embPattern_readAuto(current_pattern, argv[i+1]);
01622             printf("%d\n", current_pattern->stitch_list->count);
01623             embPattern_render(current_pattern, argv[i+2]);
01624             i += 2;
01625             break;
01626             i++;
01627             if (argv[i][0] == '-') {
01628                 /* they haven't, use the default name */
01629                 puts("Defaulting to the output name 'output.png'.");
01630                 embPattern_render(current_pattern, "output.png");
01631                 i--;
01632             }
01633             else {
01634                 /* they have, use the user-supplied name */
01635                 embPattern_render(current_pattern, argv[i]);
01636             }
01637         }
01638         else {
01639             puts("Defaulting to the output name 'output.png'.");
01640             embPattern_render(current_pattern, "output.png");
01641         }
01642         break;
01643     case FLAG_SIMULATE:
01644         if (i + 1 < argc) {
01645             /* the user appears to have entered a filename after render */
01646             i++;
01647             if (argv[i][0] == '-') {
01648                 /* they haven't, use the default name */
01649                 puts("Defaulting to the output name 'output.avi'.");
01650                 embPattern_simulate(current_pattern, "output.avi");
01651                 i--;
01652             }
01653             else {
01654                 /* they have, use the user-supplied name */
01655                 embPattern_simulate(current_pattern, argv[i]);
01656             }
01657         }
01658         else {
01659             puts("Defaulting to the output name 'output.avi'.");
01660             embPattern_simulate(current_pattern, "output.avi");
01661         }
01662         break;
01663     case FLAG_COMBINE:
01664         if (i + 3 < argc) {
01665             EmbPattern *out;
01666             EmbPattern *p1 = embPattern_create();
01667             EmbPattern *p2 = embPattern_create();
01668             embPattern_readAuto(p1, argv[i+1]);
01669             embPattern_readAuto(p2, argv[i+2]);
01670             out = embPattern_combine(p1, p2);
01671             embPattern_writeAuto(out, argv[i+3]);
01672             embPattern_free(p1);
01673             embPattern_free(p2);
01674             embPattern_free(out);
01675         }
01676         else {
01677             puts("--combine takes 3 arguments and you have supplied <3.");
01678         }
01679         break;
01680     case FLAG_VERSION:
01681     case FLAG_VERSION_SHORT:
01682         puts(version_string);
01683         break;
01684     case FLAG_HILBERT_CURVE:
01685         current_pattern = embPattern_create();
01686         hilbert_curve(current_pattern, 3);
01687         break;
01688     case FLAG_TEST:
01689         testMain(0);
01690         break;
01691     case FLAG_FULL_TEST_SUITE:
01692         testMain(1);
01693         break;
01694     case FLAG_CROSS_STITCH:
01695         if (i + 3 < argc) {
01696             EmbImage image;
01697             /* the user appears to have entered the needed arguments */
01698             image = embImage_create(2000, 2000);
01699             i++;
01700             /* to stb command */
01701             embImage_read(&image, argv[i]);
01702             i++;
01703             embPattern_crossstitch(current_pattern, &image, atoi(argv[i]));
01704             embImage_free(&image);
01705         }

```

```

01705             i++;
01706             embPattern_writeAuto(current_pattern, argv[i]);
01707         }
01708         break;
01709     default:
01710         flags--;
01711         break;
01712     }
01713 }
01714
01715 /* No flags set: default to simple from-to conversion. */
01716 if (!flags && argc == 3) {
01717     convert(argv[1], argv[2]);
01718 }
01719 else {
01720     if (!flags) {
01721         puts("Please enter an output format for your file, see --help.");
01722     }
01723 }
01724 embPattern_free(current_pattern);
01725 return 0;
01726 }
01727
01728 int main(int argc, char* argv[])
01729 {
01730     return command_line_interface(argc, argv);
01731 }
01732 #endif
01733

```

4.21 src/pattern.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "embroidery_internal.h"

```

Functions

- `EmbPattern * embPattern_create (void)`
- `void embPattern_hideStitchesOverLength (EmbPattern *p, int length)`
- `int embPattern_addThread (EmbPattern *pattern, EmbThread thread)`
- `void embPattern_fixColorCount (EmbPattern *p)`
- `void embPattern_copystitch_listToPolylines (EmbPattern *p)`
- `void embPattern_copyPolylinesToStitch_list (EmbPattern *p)`
- `void embPattern_movestitch_listToPolylines (EmbPattern *p)`
- `void embPattern_movePolylinesToStitch_list (EmbPattern *p)`
- `void embPattern_addStitchAbs (EmbPattern *p, EmbReal x, EmbReal y, int flags, int isAutoColorIndex)`
- `void embPattern_addStitchRel (EmbPattern *p, EmbReal dx, EmbReal dy, int flags, int isAutoColorIndex)`
- `void embPattern_changeColor (EmbPattern *p, int index)`
- `void embPattern_scale (EmbPattern *p, EmbReal scale)`
- `EmbRect embPattern_calcBoundingBox (EmbPattern *p)`
- `void embPattern_flipHorizontal (EmbPattern *p)`
- `void embPattern_flipVertical (EmbPattern *p)`
- `void embPattern_flip (EmbPattern *p, int horz, int vert)`
- `void embPattern_combineJumpStitches (EmbPattern *p)`
- `void embPattern_correctForMaxStitchLength (EmbPattern *p, EmbReal maxStitchLength, EmbReal maxJumpLength)`
- `void embPattern_center (EmbPattern *p)`
- `void embPattern_loadExternalColorFile (EmbPattern *p, const char *fileName)`

- void `embPattern_free (EmbPattern *p)`
- void `embPattern_addCircleAbs (EmbPattern *p, EmbCircle circle)`
- void `embPattern_addEllipseAbs (EmbPattern *p, EmbEllipse ellipse)`
- void `embPattern_addLineAbs (EmbPattern *p, EmbLine line)`
- void `embPattern_addPathAbs (EmbPattern *p, EmbPath obj)`
- void `embPattern_addPointAbs (EmbPattern *p, EmbPoint obj)`
- void `embPattern_addPolygonAbs (EmbPattern *p, EmbPolygon obj)`
- void `embPattern_addPolylineObjectAbs (EmbPattern *p, EmbPolyline obj)`
- void `embPattern_addRectAbs (EmbPattern *p, EmbRect rect)`
- void `embPattern_end (EmbPattern *p)`
- int `embPattern_color_count (EmbPattern *pattern, EmbColor startColor)`
- void `embPattern_designDetails (EmbPattern *pattern)`
- int `convert (const char *inf, const char *outf)`
- float `embPattern_totalStitchLength (EmbPattern *pattern)`
- float `embPattern_minimumStitchLength (EmbPattern *pattern)`
- float `embPattern_maximumStitchLength (EmbPattern *pattern)`
- void `embPattern_lengthHistogram (EmbPattern *pattern, int *bin, int NUMBINS)`
- int `embPattern_realStitches (EmbPattern *pattern)`
- int `embPattern_jumpStitches (EmbPattern *pattern)`
- int `embPattern_trimStitches (EmbPattern *pattern)`

4.21.1 Function Documentation

4.21.1.1 convert()

```
int convert (
    const char * inf,
    const char * outf )
```

Definition at line 1108 of file [pattern.c](#).

4.21.1.2 embPattern_addCircleAbs()

```
void embPattern_addCircleAbs (
    EmbPattern * p,
    EmbCircle circle )
```

Adds a circle object to pattern (*p*) with its center at the absolute position (*cx, cy*) with a radius of (*r*). Positive y is up.
Units are in millimeters.

Definition at line 787 of file [pattern.c](#).

4.21.1.3 embPattern_addEllipseAbs()

```
void embPattern_addEllipseAbs (
    EmbPattern * p,
    EmbEllipse ellipse )
```

Adds an ellipse object to pattern (*p*) with its center at the absolute position (*cx,cy*) with radii of (*rx,ry*). Positive y is up. Units are in millimeters.

Definition at line 801 of file [pattern.c](#).

4.21.1.4 embPattern_addLineAbs()

```
void embPattern_addLineAbs (
    EmbPattern * p,
    EmbLine line )
```

Adds a line object to pattern (*p*) starting at the absolute position (*x1,y1*) and ending at the absolute position (*x2,y2*). Positive y is up. Units are in millimeters.

Definition at line 816 of file [pattern.c](#).

4.21.1.5 embPattern_addPathAbs()

```
void embPattern_addPathAbs (
    EmbPattern * p,
    EmbPath obj )
```

Definition at line 827 of file [pattern.c](#).

4.21.1.6 embPattern_addPointAbs()

```
void embPattern_addPointAbs (
    EmbPattern * p,
    EmbPoint obj )
```

Adds a point object to pattern (*p*) at the absolute position (*x,y*). Positive y is up. Units are in millimeters.

Definition at line 843 of file [pattern.c](#).

4.21.1.7 embPattern_addPolygonAbs()

```
void embPattern_addPolygonAbs (
    EmbPattern * p,
    EmbPolygon obj )
```

Definition at line [854](#) of file [pattern.c](#).

4.21.1.8 embPattern_addPolylineObjectAbs()

```
void embPattern_addPolylineObjectAbs (
    EmbPattern * p,
    EmbPolyline obj )
```

Definition at line [869](#) of file [pattern.c](#).

4.21.1.9 embPattern_addRectAbs()

```
void embPattern_addRectAbs (
    EmbPattern * p,
    EmbRect rect )
```

Adds a rectangle object to pattern (*p*) at the absolute position (*x,y*) with a width of (*w*) and a height of (*h*). Positive y is up. Units are in millimeters.

Definition at line [888](#) of file [pattern.c](#).

4.21.1.10 embPattern_addStitchAbs()

```
void embPattern_addStitchAbs (
    EmbPattern * p,
    EmbReal x,
    EmbReal y,
    int flags,
    int isAutoColorIndex )
```

Adds a stitch to the pattern (*p*) at the absolute position (*x,y*). Positive y is up. Units are in millimeters.

Definition at line [236](#) of file [pattern.c](#).

4.21.1.11 embPattern_addStitchRel()

```
void embPattern_addStitchRel (
    EmbPattern * p,
    EmbReal dx,
    EmbReal dy,
    int flags,
    int isAutoColorIndex )
```

Adds a stitch to the pattern (*p*) at the relative position (*dx,dy*) to the previous stitch. Positive y is up. Units are in millimeters.

Definition at line 290 of file [pattern.c](#).

4.21.1.12 embPattern_addThread()

```
int embPattern_addThread (
    EmbPattern * pattern,
    EmbThread thread )
```

Definition at line 66 of file [pattern.c](#).

4.21.1.13 embPattern_calcBoundingBox()

```
EmbRect embPattern_calcBoundingBox (
    EmbPattern * p )
```

Returns an EmbRect that encapsulates all stitches and objects in the pattern (*p*).

Definition at line 340 of file [pattern.c](#).

4.21.1.14 embPattern_center()

```
void embPattern_center (
    EmbPattern * p )
```

Definition at line 709 of file [pattern.c](#).

4.21.1.15 embPattern_changeColor()

```
void embPattern_changeColor (
    EmbPattern * p,
    int index )
```

Definition at line 312 of file [pattern.c](#).

4.21.1.16 embPattern_color_count()

```
int embPattern_color_count (
    EmbPattern * pattern,
    EmbColor startColor )
```

Definition at line [911](#) of file [pattern.c](#).

4.21.1.17 embPattern_combineJumpStitches()

```
void embPattern_combineJumpStitches (
    EmbPattern * p )
```

Definition at line [624](#) of file [pattern.c](#).

4.21.1.18 embPattern_copyPolylinesToStitch_list()

```
void embPattern_copyPolylinesToStitch_list (
    EmbPattern * p )
```

Copies all of the EmbPolylineObjectList data to Embstitch_list data for pattern (*p*).

Definition at line [161](#) of file [pattern.c](#).

4.21.1.19 embPattern_copyStitch_listToPolylines()

```
void embPattern_copyStitch_listToPolylines (
    EmbPattern * p )
```

Copies all of the Embstitch_list data to EmbPolylineObjectList data for pattern (*p*).

Definition at line [114](#) of file [pattern.c](#).

4.21.1.20 embPattern_correctForMaxStitchLength()

```
void embPattern_correctForMaxStitchLength (
    EmbPattern * p,
    EmbReal maxStitchLength,
    EmbReal maxJumpLength )
```

Definition at line [660](#) of file [pattern.c](#).

4.21.1.21 embPattern_create()

```
EmbPattern * embPattern_create (
    void )
```

Returns a pointer to an EmbPattern. It is created on the heap. The caller is responsible for freeing the allocated memory with [embPattern_free\(\)](#).

Definition at line 21 of file [pattern.c](#).

4.21.1.22 embPattern_designDetails()

```
void embPattern_designDetails (
    EmbPattern * pattern )
```

Definition at line 942 of file [pattern.c](#).

4.21.1.23 embPattern_end()

```
void embPattern_end (
    EmbPattern * p )
```

Definition at line 898 of file [pattern.c](#).

4.21.1.24 embPattern_fixColorCount()

```
void embPattern_fixColorCount (
    EmbPattern * p )
```

Definition at line 79 of file [pattern.c](#).

4.21.1.25 embPattern_flip()

```
void embPattern_flip (
    EmbPattern * p,
    int horz,
    int vert )
```

Flips the entire pattern (*p*) horizontally about the x-axis if (*horz*) is true. Flips the entire pattern (*p*) vertically about the y-axis if (*vert*) is true.

Definition at line 497 of file [pattern.c](#).

4.21.1.26 `embPattern_flipHorizontal()`

```
void embPattern_flipHorizontal (
    EmbPattern * p )
```

Flips the entire pattern (*p*) horizontally about the y-axis.

Definition at line [472](#) of file [pattern.c](#).

4.21.1.27 `embPattern_flipVertical()`

```
void embPattern_flipVertical (
    EmbPattern * p )
```

Flips the entire pattern (*p*) vertically about the x-axis.

Definition at line [484](#) of file [pattern.c](#).

4.21.1.28 `embPattern_free()`

```
void embPattern_free (
    EmbPattern * p )
```

Frees all memory allocated in the pattern (*p*).

Definition at line [771](#) of file [pattern.c](#).

4.21.1.29 `embPattern_hideStitchesOverLength()`

```
void embPattern_hideStitchesOverLength (
    EmbPattern * p,
    int length )
```

Definition at line [42](#) of file [pattern.c](#).

4.21.1.30 `embPattern_jumpStitches()`

```
int embPattern_jumpStitches (
    EmbPattern * pattern )
```

Definition at line [1240](#) of file [pattern.c](#).

4.21.1.31 embPattern_lengthHistogram()

```
void embPattern_lengthHistogram (
    EmbPattern * pattern,
    int * bin,
    int NUMBINS )
```

Definition at line [1205](#) of file [pattern.c](#).

4.21.1.32 embPattern_loadExternalColorFile()

```
void embPattern_loadExternalColorFile (
    EmbPattern * p,
    const char * fileName )
```

Definition at line [732](#) of file [pattern.c](#).

4.21.1.33 embPattern_maximumStitchLength()

```
float embPattern_maximumStitchLength (
    EmbPattern * pattern )
```

Definition at line [1184](#) of file [pattern.c](#).

4.21.1.34 embPattern_minimumStitchLength()

```
float embPattern_minimumStitchLength (
    EmbPattern * pattern )
```

Definition at line [1163](#) of file [pattern.c](#).

4.21.1.35 embPattern_movePolylinesTostitch_list()

```
void embPattern_movePolylinesTostitch_list (
    EmbPattern * p )
```

Moves all of the EmbPolylineObjectList data to Embstitch_list data for pattern (*p*).

Definition at line [225](#) of file [pattern.c](#).

4.21.1.36 `embPattern_movestitch_listToPolylines()`

```
void embPattern_movestitch_listToPolylines (
    EmbPattern * p )
```

Moves all of the Embstitch_list data to EmbPolylineObjectList data for pattern (*p*).

Definition at line 211 of file [pattern.c](#).

4.21.1.37 `embPattern_realStitches()`

```
int embPattern_realStitches (
    EmbPattern * pattern )
```

Definition at line 1227 of file [pattern.c](#).

4.21.1.38 `embPattern_scale()`

```
void embPattern_scale (
    EmbPattern * p,
    EmbReal scale )
```

Definition at line 324 of file [pattern.c](#).

4.21.1.39 `embPattern_totalStitchLength()`

```
float embPattern_totalStitchLength (
    EmbPattern * pattern )
```

Definition at line 1143 of file [pattern.c](#).

4.21.1.40 `embPattern_trimStitches()`

```
int embPattern_trimStitches (
    EmbPattern * pattern )
```

Definition at line 1254 of file [pattern.c](#).

4.22 pattern.c

[Go to the documentation of this file.](#)

```

00001 /*
00002 * This file is part of libembroidery.
00003 *
00004 * Copyright 2018-2022 The Embroidermodder Team
00005 * Licensed under the terms of the zlib license.
00006 *
00007 * The file is for the management of the main struct: EmbPattern.
00008 ****
00009
00010 #include <stdio.h>
00011 #include <stdlib.h>
00012 #include <string.h>
00013 #include <math.h>
00014
00015 #include "embroidery_internal.h"
00016
00020 EmbPattern*
00021 embPattern_create(void)
00022 {
00023     EmbPattern* p = (EmbPattern*)malloc(sizeof(EmbPattern));
00024     if (!p) {
00025         printf("ERROR: emb-pattern.c embPattern_create(), ");
00026         printf("unable to allocate memory for p\n");
00027         return 0;
00028     }
00029     p->dstJumpsPerTrim = 6;
00030     p->home.x = 0.0;
00031     p->home.y = 0.0;
00032     p->currentColorIndex = 0;
00033     p->stitch_list = embArray_create(EMB_STITCH);
00034     p->thread_list = embArray_create(EMB_THREAD);
00035     p->hoop_height = 0.0;
00036     p->hoop_width = 0.0;
00037     p->geometry = embArray_create(EMB_LINE);
00038     return p;
00039 }
00040
00041 void
00042 embPattern_hideStitchesOverLength(EmbPattern* p, int length)
00043 {
00044     EmbVector prev;
00045     int i;
00046     prev.x = 0.0;
00047     prev.y = 0.0;
00048
00049     if (!p) {
00050         printf("ERROR: emb-pattern.c embPattern_hideStitchesOverLength(), ");
00051         printf("p argument is null\n");
00052         return;
00053     }
00054     for (i = 0; i < p->stitch_list->count; i++) {
00055         if ((fabs(p->stitch_list->stitch[i].x - prev.x) > length)
00056             || (fabs(p->stitch_list->stitch[i].y - prev.y) > length)) {
00057             p->stitch_list->stitch[i].flags |= TRIM;
00058             p->stitch_list->stitch[i].flags &= ~NORMAL;
00059         }
00060         prev.x = p->stitch_list->stitch[i].x;
00061         prev.y = p->stitch_list->stitch[i].y;
00062     }
00063 }
00064
00065 int
00066 embPattern_addThread(EmbPattern *pattern, EmbThread thread)
00067 {
00068     if (pattern->thread_list->count + 1 > pattern->thread_list->length) {
00069         if (!embArray_resize(pattern->thread_list)) {
00070             return 0;
00071         }
00072     }
00073     pattern->thread_list->thread[pattern->thread_list->count] = thread;
00074     pattern->thread_list->count++;
00075     return 1;
00076 }
00077
00078 void
00079 embPattern_fixColorCount(EmbPattern* p)
00080 {
00081     /* fix color count to be max of color index. */
00082     int maxColorIndex = 0, i;
00083
00084     if (!p) {
00085         printf("ERROR: emb-pattern.c embPattern_fixColorCount(), ");
00086     }

```

```

00086     printf("p argument is null\n");
00087     return;
00088 }
00089 for (i = 0; i < p->stitch_list->count; i++) {
00090 /*     printf("%d %d\n", list->stitch.color, maxColorIndex); */
00091     maxColorIndex = EMB_MAX(maxColorIndex, p->stitch_list->stitch[i].color);
00092 }
00093 if (p->thread_list->count == 0 || maxColorIndex == 0) {
00094     embPattern_addThread(p, black_thread);
00095 }
00096 else {
00097     if (maxColorIndex > 0) {
00098         while (p->thread_list->count <= maxColorIndex) {
00099 /*         printf("%d %d\n", p->n_threads, maxColorIndex); */
00100         embPattern_addThread(p, embThread_getRandom());
00101     }
00102 }
00103 }
00104 /*
00105 while (p->threadLists->count > (maxColorIndex + 1)) {
00106     TODO: erase last color    p->threadList.pop_back();
00107 }
00108 */
00109 }
00110
00111 void
00112 embPattern_copystitch_listToPolylines(EmbPattern* p)
00113 {
00114     int breakAtFlags, i;
00115     EmbPoint point;
00116     EmbColor color;
00117
00118     if (!p) {
00119         printf("ERROR: emb-pattern.c embPattern_copystitch_listToPolylines(), ");
00120         printf("p argument is null\n");
00121         return;
00122     }
00123     breakAtFlags = (STOP | JUMP | TRIM);
00124
00125     for (i = 0; i < p->stitch_list->count; i++) {
00126         EmbArray *pointList = 0;
00127         for (; i < p->stitch_list->count; i++) {
00128             EmbStitch st = p->stitch_list->stitch[i];
00129             if (st.flags & breakAtFlags) {
00130                 break;
00131             }
00132             if (!(st.flags & JUMP)) {
00133                 if (!pointList) {
00134                     pointList = embArray_create(EMB_POINT);
00135                     color = p->thread_list->thread[st.color].color;
00136                 }
00137                 point.position.x = st.x;
00138                 point.position.y = st.y;
00139                 embArray_addPoint(pointList, point);
00140             }
00141         }
00142     }
00143
00144     /* NOTE: Ensure empty polylines are not created. This is critical. */
00145     if (pointList) {
00146         EmbPolyline currentPolyline;
00147         currentPolyline.pointList = pointList;
00148         currentPolyline.color = color;
00149         /* TODO: Determine what the correct value should be */
00150         currentPolyline.lineType = 1;
00151
00152         embArray_addPolyline(p->geometry, currentPolyline);
00153     }
00154 }
00155 }
00156 }
00157 }
00158
00159 void
00160 embPattern_copyPolylinesTostitch_list(EmbPattern* p)
00161 {
00162     int firstObject = 1, i, j;
00163     /*int currentColor = polylineObj->color TODO: polyline color */
00164
00165     if (!p) {
00166         printf("ERROR: emb-pattern.c embPattern_copyPolylinesTostitch_list(), ");
00167         printf("p argument is null\n");
00168         return;
00169     }
00170     for (i = 0; i < p->geometry->count; i++) {
00171         EmbPolyline currentPoly;
00172         EmbArray* currentPointList;
00173         EmbThread thread;
00174
00175 }
```

```

00176     if (p->geometry->geometry[i].type != EMB_POLYLINE) {
00177         continue;
00178     }
00179
00180     currentPoly = p->geometry->geometry[i].object.polyline;
00181     currentPointList = currentPoly.pointList;
00182
00183     strcpy(thread.catalogNumber, "");
00184     thread.color = currentPoly.color;
00185     strcpy(thread.description, "");
00186     embPattern_addThread(p, thread);
00187
00188     if (!firstObject) {
00189         embPattern_addStitchAbs(p,
00190             currentPointList->geometry[0].object.point.position.x,
00191             currentPointList->geometry[0].object.point.position.y, TRIM, 1);
00192         embPattern_addStitchRel(p, 0.0, 0.0, STOP, 1);
00193     }
00194
00195     embPattern_addStitchAbs(p,
00196         currentPointList->geometry[0].object.point.position.x,
00197         currentPointList->geometry[0].object.point.position.y,
00198         JUMP,
00199         1);
00200     for (j = 1; j < currentPointList->count; j++) {
00201         EmbVector v = currentPointList->geometry[j].object.point.position;
00202         embPattern_addStitchAbs(p, v.x, v.y, NORMAL, 1);
00203     }
00204     firstObject = 0;
00205 }
00206 embPattern_addStitchRel(p, 0.0, 0.0, END, 1);
00207 }
00208
00209 void
00210 embPattern_movestitch_listToPolylines(EmbPattern* p)
00211 {
00212     if (!p) {
00213         printf("ERROR: emb-pattern.c embPattern_movestitch_listToPolylines(), p argument is null\n");
00214         return;
00215     }
00216     embPattern_copystitch_listToPolylines(p);
00217     /* Free the stitch_list and threadList since their data has now been transferred to polylines */
00218     p->stitch_list->count = 0;
00219     p->thread_list->count = 0;
00220 }
00221 }
00222
00223 void
00224 embPattern_movePolylinesTostitch_list(EmbPattern* p)
00225 {
00226     if (!p) {
00227         printf("ERROR: emb-pattern.c embPattern_movePolylinesTostitch_list(), p argument is null\n");
00228         return;
00229     }
00230     embPattern_copyPolylinesTostitch_list(p);
00231 }
00232 }
00233
00234 void
00235 embPattern_addStitchAbs(EmbPattern* p, EmbReal x, EmbReal y,
00236                           int flags, int isAutoColorIndex)
00237 {
00238     EmbStitch s;
00239
00240     if (!p) {
00241         printf("ERROR: emb-pattern.c embPattern_addStitchAbs(), p argument is null\n");
00242         printf("p argument is null\n");
00243         return;
00244     }
00245
00246     if (flags & END) {
00247         if (p->stitch_list->count == 0) {
00248             return;
00249         }
00250         /* Prevent unnecessary multiple END stitches */
00251         if (p->stitch_list->stitch[p->stitch_list->count - 1].flags & END) {
00252             printf("ERROR: emb-pattern.c embPattern_addStitchAbs(), found multiple END stitches\n");
00253             return;
00254         }
00255         embPattern_fixColorCount(p);
00256         /* HideStitchesOverLength(127); TODO: fix or remove this */
00257     }
00258
00259     if (flags & STOP) {
00260         if (p->stitch_list->count == 0) {
00261             return;
00262         }
00263         if (isAutoColorIndex) {
00264             p->currentColorIndex++;

```

```

00266         }
00267     }
00268
00269     /* NOTE: If the stitch_list is empty, we will create it before adding
00270     stitches to it. The first coordinate will be the HOME position. */
00271     if (p->stitch_list->count == 0) {
00272         /* NOTE: Always HOME the machine before starting any stitching */
00273         EmbStitch h;
00274         h.x = p->home.x;
00275         h.y = p->home.y;
00276         h.flags = JUMP;
00277         h.color = p->currentColorIndex;
00278         embArray_addStitch(p->stitch_list, h);
00279     }
00280     s.x = x;
00281     s.y = y;
00282     s.flags = flags;
00283     s.color = p->currentColorIndex;
00284     embArray_addStitch(p->stitch_list, s);
00285 }
00286
00287 void
00288 embPattern_addStitchRel(EmbPattern* p, EmbReal dx, EmbReal dy,
00289                         int flags, int isAutoColorIndex)
00290 {
00291     EmbReal x, y;
00292     if (!p) {
00293         printf("ERROR: emb-pattern.c embPattern_addStitchRel(), p argument is null\n");
00294         return;
00295     }
00296     if (p->stitch_list->count > 0) {
00297         EmbStitch st = p->stitch_list->stitch[p->stitch_list->count - 1];
00298         x = st.x + dx;
00299         y = st.y + dy;
00300     } else {
00301         /* NOTE: The stitch_list is empty, so add it to the HOME position.
00302         * The embstitch_list_create function will ensure the first coordinate is at the HOME
00303         position. */
00304         x = p->home.x + dx;
00305         y = p->home.y + dy;
00306     }
00307     embPattern_addStitchAbs(p, x, y, flags, isAutoColorIndex);
00308 }
00309
00310
00311 void
00312 embPattern_changeColor(EmbPattern* p, int index)
00313 {
00314     if (!p) {
00315         printf("ERROR: emb-pattern.c embPattern_changeColor(), p argument is null\n");
00316         return;
00317     }
00318     p->currentColorIndex = index;
00319 }
00320
00321 /* Very simple scaling of the x and y axis for every point.
00322 * Doesn't insert or delete stitches to preserve density. */
00323 void
00324 embPattern_scale(EmbPattern* p, EmbReal scale)
00325 {
00326     int i;
00327     if (!p) {
00328         printf("ERROR: emb-pattern.c embPattern_scale(), p argument is null\n");
00329         return;
00330     }
00331
00332     for (i = 0; i < p->stitch_list->count; i++) {
00333         p->stitch_list->stitch[i].x *= scale;
00334         p->stitch_list->stitch[i].y *= scale;
00335     }
00336 }
00337
00338 EmbRect
00339 embPattern_calcBoundingBox(EmbPattern* p)
00340 {
00341     EmbRect r;
00342     EmbStitch pt;
00343     int i, j;
00344
00345     r.left = 0;
00346     r.right = 0;
00347     r.top = 0;
00348     r.bottom = 0;
00349
00350     if (!p) {
00351         printf("ERROR: emb-pattern.c embPattern_calcBoundingBox(), ");
00352         printf("p argument is null\n");
00353         return r;
00354     }

```

```

00355     }
00356
00357     /* Calculate the bounding rectangle. It's needed for smart repainting. */
00358     /* TODO: Come back and optimize this mess so that after going thru all objects
00359      and stitches, if the rectangle isn't reasonable, then return a default rect */
00360     if ((p->stitch_list->count == 0) && (p->geometry->count == 0)) {
00361         r.top = 0.0;
00362         r.left = 0.0;
00363         r.bottom = 1.0;
00364         r.right = 1.0;
00365         return r;
00366     }
00367     r.left = 99999.0;
00368     r.top = 99999.0;
00369     r.right = -99999.0;
00370     r.bottom = -99999.0;
00371
00372     for (i = 0; i < p->stitch_list->count; i++) {
00373         /* If the point lies outside of the accumulated bounding
00374          * rectangle, then inflate the bounding rect to include it. */
00375         pt = p->stitch_list->stitch[i];
00376         if (!(pt.flags & TRIM)) {
00377             r.left = EMB_MIN(r.left, pt.x);
00378             r.top = EMB_MIN(r.top, pt.y);
00379             r.right = EMB_MAX(r.right, pt.x);
00380             r.bottom = EMB_MAX(r.bottom, pt.y);
00381         }
00382     }
00383
00384     for (i = 0; i < p->geometry->count; i++) {
00385         EmbGeometry g = p->geometry->geometry[i];
00386         switch (g.type) {
00387             case EMB_ARC: {
00388                 /* TODO: embPattern_calcBoundingBox for arcs,
00389                  for now just checks the start point */
00390                 EmbArc arc = g.object.arc;
00391                 r.left = EMB_MIN(r.left, arc.start.x);
00392                 r.top = EMB_MIN(r.top, arc.start.y);
00393                 r.right = EMB_MAX(r.right, arc.start.x);
00394                 r.bottom = EMB_MAX(r.bottom, arc.start.y);
00395                 break;
00396             }
00397             case EMB_CIRCLE: {
00398                 EmbCircle circle = g.object.circle;
00399                 r.left = EMB_MIN(r.left, circle.center.x - circle.radius);
00400                 r.top = EMB_MIN(r.top, circle.center.y - circle.radius);
00401                 r.right = EMB_MAX(r.right, circle.center.x + circle.radius);
00402                 r.bottom = EMB_MAX(r.bottom, circle.center.y + circle.radius);
00403                 break;
00404             }
00405             case EMB_ELLIPSE: {
00406                 /* TODO: account for rotation */
00407                 EmbEllipse ellipse = g.object.ellipse;
00408                 r.left = EMB_MIN(r.left, ellipse.center.x - ellipse.radius.x);
00409                 r.top = EMB_MIN(r.top, ellipse.center.y - ellipse.radius.y);
00410                 r.right = EMB_MAX(r.right, ellipse.center.x + ellipse.radius.x);
00411                 r.bottom = EMB_MAX(r.bottom, ellipse.center.y + ellipse.radius.y);
00412                 break;
00413             }
00414             case EMB_LINE: {
00415                 EmbLine line = g.object.line;
00416                 r.left = EMB_MIN(r.left, line.start.x);
00417                 r.left = EMB_MIN(r.left, line.end.x);
00418                 r.top = EMB_MIN(r.top, line.start.y);
00419                 r.top = EMB_MIN(r.top, line.end.y);
00420                 r.right = EMB_MAX(r.right, line.start.x);
00421                 r.right = EMB_MAX(r.right, line.end.x);
00422                 r.bottom = EMB_MAX(r.bottom, line.start.y);
00423                 r.bottom = EMB_MAX(r.bottom, line.end.y);
00424                 break;
00425             }
00426             case EMB_POINT: {
00427                 EmbVector point = g.object.point.position;
00428                 r.left = EMB_MIN(r.left, point.x);
00429                 r.top = EMB_MIN(r.top, point.y);
00430                 r.right = EMB_MAX(r.right, point.x);
00431                 r.bottom = EMB_MAX(r.bottom, point.y);
00432                 break;
00433             }
00434             case EMB_POLYGON: {
00435                 EmbArray *polygon = g.object.polygon.pointList;
00436                 for (j=0; j < polygon->count; j++) {
00437                     /* TODO: embPattern_calcBoundingBox for polygons */
00438                 }
00439                 break;
00440             }
00441             case EMB_POLYLINE: {

```

```

00442         EmbArray *polyline = g->object.polyline.pointList;
00443         for (j=0; j < polyline->count; j++) {
00444             /* TODO: embPattern_calcBoundingBox for polylines */
00445         }
00446         break;
00447     }
00448     case EMB_RECT: {
00449         EmbRect rect = g->object.rect;
00450         r.left = EMB_MIN(r.left, rect.left);
00451         r.top = EMB_MIN(r.top, rect.top);
00452         r.right = EMB_MAX(r.right, rect.right);
00453         r.bottom = EMB_MAX(r.bottom, rect.bottom);
00454         break;
00455     }
00456     case EMB_SPLINE: {
00457         /* EmbBezier bezier;
00458         bezier = p->splines->spline[i].bezier; */
00459         /* TODO: embPattern_calcBoundingBox for splines */
00460         break;
00461     }
00462     default:
00463         break;
00464     }
00465 }
00466
00467     return r;
00468 }
00469
00470 void
00471 embPattern_flipHorizontal(EmbPattern* p)
00472 {
00473     if (!p) {
00474         printf("ERROR: emb-pattern.c embPattern_flipHorizontal(), ");
00475         printf("p argument is null\n");
00476         return;
00477     }
00478     embPattern_flip(p, 1, 0);
00479 }
00480
00481
00482 void
00483 embPattern_flipVertical(EmbPattern* p)
00484 {
00485     if (!p) {
00486         printf("ERROR: emb-pattern.c embPattern_flipVertical(), ");
00487         printf("p argument is null\n");
00488         return;
00489     }
00490     embPattern_flip(p, 0, 1);
00491 }
00492
00493
00494 void
00495 embPattern_flip(EmbPattern* p, int horz, int vert)
00496 {
00497     int i, j;
00498
00499     if (!p) {
00500         printf("ERROR: emb-pattern.c embPattern_flip(), p argument is null\n");
00501         return;
00502     }
00503
00504     for (i = 0; i < p->stitch_list->count; i++) {
00505         if (horz) {
00506             p->stitch_list->stitch[i].x *= -1.0;
00507         }
00508         if (vert) {
00509             p->stitch_list->stitch[i].y *= -1.0;
00510         }
00511     }
00512
00513 }
00514
00515     for (i = 0; i < p->geometry->count; i++) {
00516         EmbGeometry *g = &(p->geometry->geometry[i]);
00517         switch (g->type) {
00518             case EMB_ARC: {
00519                 if (horz) {
00520                     g->object.arc.start.x *= -1.0;
00521                     g->object.arc.mid.x *= -1.0;
00522                     g->object.arc.end.x *= -1.0;
00523                 }
00524                 if (vert) {
00525                     g->object.arc.start.y *= -1.0;
00526                     g->object.arc.mid.y *= -1.0;
00527                     g->object.arc.end.y *= -1.0;
00528                 }
00529                 break;
00530             }
00531             case EMB_LINE: {
00532                 if (horz) {
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
010010
010011
010012
010013
010014
010015
010016
010017
010018
010019
010020
010021
010022
010023
010024
010025
010026
010027
010028
010029
010030
010031
010032
010033
010034
010035
010036
010037
010038
010039
010040
010041
010042
010043
010044
010045
010046
010047
010048
010049
010050
010051
010052
010053
010054
010055
010056
010057
010058
010059
010060
010061
010062
010063
010064
010065
010066
010067
010068
010069
010070
010071
010072
010073
010074
010075
010076
010077
010078
010079
010080
010081
010082
010083
010084
010085
010086
010087
010088
010089
010090
010091
010092
010093
010094
010095
010096
010097
010098
010099
0100100
0100101
0100102
0100103
0100104
0100105
0100106
0100107
0100108
0100109
0100110
0100111
0100112
0100113
0100114
0100115
0100116
0100117
0100118
0100119
0100120
0100121
0100122
0100123
0100124
0100125
0100126
0100127
0100128
0100129
0100130
0100131
0100132
0100133
0100134
0100135
0100136
0100137
0100138
0100139
0100140
0100141
0100142
0100143
0100144
0100145
0100146
0100147
0100148
0100149
0100150
0100151
0100152
0100153
0100154
0100155
0100156
0100157
0100158
0100159
0100160
0100161
0100162
0100163
0100164
0100165
0100166
0100167
0100168
0100169
0100170
0100171
0100172
0100173
0100174
0100175
0100176
0100177
0100178
0100179
0100180
0100181
0100182
0100183
0100184
0100185
0100186
0100187
0100188
0100189
0100190
0100191
0100192
0100193
0100194
0100195
0100196
0100197
0100198
0100199
0100200
0100201
0100202
0100203
0100204
0100205
0100206
0100207
0100208
0100209
0100210
0100211
0100212
0100213
0100214
0100215
0100216
0100217
0100218
0100219
0100220
0100221
0100222
0100223
0100224
0100225
0100226
0100227
0100228
0100229
0100230
0100231
0100232
0100233
0100234
0100235
0100236
0100237
0100238
0100239
0100240
0100241
0100242
0100243
0100244
0100245
0100246
0100247
0100248
0100249
0100250
0100251
0100252
0100253
0100254
0100255
0100256
0100257
0100258
0100259
0100260
0100261
0100262
0100263
0100264
0100265
0100266
0100267
0100268
0100269
0100270
0100271
0100272
0100273
0100274
0100275
0100276
0100277
0100278
0100279
0100280
0100281
0100282
0100283
0100284
0100285
0100286
0100287
0100288
0100289
0100290
0100291
0100292
0100293
0100294
0100295
0100296
0100297
0100298
0100299
0100300
0100301
0100302
0100303
0100304
0100305
0100306
0100307
0100308
0100309
0100310
0100311
0100312
0100313
0100314
0100315
0100316
0100317
0100318
0100319
0100320
0100321
0100322
0100323
0100324
0100325
0100326
0100327
0100328
0100329
0100330
0100331
0100332
0100333
0100334
0100335
0100336
0100337
0100338
0100339
0100340
0100341
0100342
0100343
0100344
0100345
0100346
0100347
0100348
0100349
0100350
0100351
0100352
0100353
0100354
0100355
0100356
0100357
0100358
0100359
0100360
0100361
0100362
0100363
0100364
0100365
0100366
0100367
0100368
0100369
0100370
0100371
0100372
0100373
0100374
0100375
0100376
0100377
0100378
0100379
0100380
0100381
0100382
0100383
0100384
0100385
0100386
0100387
0100388
0100389
0100390
0100391
0100392
0100393
0100394
0100395
0100396
0100397
0100398
0100399
0100400
0100401
0100402
0100403
0100404
0100405
0100406
0100407
0100408
0100409
0100410
0100411
0100412
0100413
0100414
0100415
0100416
0100417
0100418
0100419
0100420
0100421
0100422
0100423
0100424
0100425
0100426
0100427
0100428
0100429
0100430
0100431
0100432
0100433
0100434
0100435
0100436
0100437
0100438
0100439
0100440
0100441
0100442
0100443
0100444
0100445
0100446
0100447
0100448
0100449
0100450
0100451
0100452
0100453
0100454
0100455
0100456
0100457
0100458
0100459
0100460
0100461
0100462
0100463
0100464
0100465
0100466
0100467
0100468
0100469
0100470
0100471
0100472
0100473
0100474
0100475
0100476
0100477
0100478
0100479
0100480
0100481
0100482
0100483
0100484
0100485
0100486
0100487
0100488
0100489
0100490
0100491
0100492
0100493
0100494
0100495
0100496
0100497
0100498
0100499
0100500
0100501
0100502
0100503
0100504
0100505
0100506
0100507
0100508
0100509
0100510
0100511
0100512
0100513
0100514
0100515
0100516
0100517
0100518
0100519
0100520
0100521
0100522
0100523
0100524
0100525
0100526
0100527
0100528
0100529
0100530
0100531
0100532
0100533
0100534
0100535
0100536
0100537
0100538
0100539
0100540
0100541
0100542
0100543
0100544
0100545
0100546
0100547
0100548
0100549
0100550
0100551
0100552
0100553
0100554
0100555
0100556
0100557
0100558
0100559
0100560
0100561
0100562
0100563
0100564
0100565
0100566
0100567
0100568
0100569
0100570
0100571
0100572
0100573
0100574
0100575
0100576
0100577
0100578
0100579
0100580
0100581
0100582
0100583
0100584
0100585
0100586
0100587
0100588
0100589
0100590
0100591
0100592
0100593
0100594
0100595
0100596
0100597
0100598
0100599
0100600
0100601
0100602
0100603
0100604
0100605
0100606
0100607
0100608
0100609
0100610
0100611
0100612
0100613
0100614
0100615
0100616
0100617
0100618
0100619
0100620
0100621
0100622
0100623
0100624
0100625
0100626
0100627
0100628
0100629
0100630
0100631
0100632
0100633
0100634
0100635
0100636
0100637
0100638
0100639
0100640
0100641
0100642
0100643
0100644
0100645
0100646
0100647
0100648
0100649
0100650
0100651
0100652
0100653
0100654
0100655
0100656
0100657
0100658
0100659
0100660
0100661
0100662
0100663
0100664
0100665
0100666
0100667
0100668
0100669
0100670
0100671
0100672
0100673
0100674
0100675
0100676
0100677
0100678
0100679
0100680
0100681
0100682
0100683
0100684
0100685
0100686
0100687
0100688
0100689
0100690
0100691
0100692
0100693
0100694
0100695
0100696
0100697
0100698
0100699
0100700
0100701
0100702
0100703
0100704
0100705
0100706
0100707
0100708
0100709
0100710
0100711
0100712
0100713
0100714
0100715
0100716
0100717
0100718
0100719
0100720
0100721
0100722
0100723
0100724
0100725
0100726
0100727
0100728
0100729
0100730
0100731
0100732
0100733
0100734
0100735
0100736
0100737
0100738
0100739
0100740
0100741
0100742
0100743
0100744
0100745
0100746
0100747
0100748
0100749
0100750
0100751
0100752
0100753
0100754
0100755
0100756
0100757
0100758
0100759
0100760
0100761
0100762
0100763
0100764
0100765
0100766
0100767
0100768
0100769
0100770
0100771
0100772
0100773
0100774
0100775
0100776
0100777
0100778
0100779
0100780
0100781
0100782
0100783
0100784
0100785
0100786
0100787
0100788
0100789
0100790
0100791
0100792
0100793
0100794
0100795
0100796
0100797
0100798
0100799
0100800
0100801
0100802
0100803
0100804
0100805
0100806
0100807
0100808
0100809
0100810
0100811
0100812
0100813
0100814
0100815
0100816
0100817
0100818
0100819
0100820
0100821
0100822
0100823
0100824
0100825
0100826
0100827
0100828
0100829
0100830
0100831
0100832
0100833
0100834
0100835
0100836
0100837
0100838
0100839
0100840
0100841
0100842
0100843
0100844
0100845
0100846
0100847
0100848
0100849
0100850
0100851
0100852
0100853
0100854
0100855
0100856
0100857
0100858
0100859
0100860
0100861
0100862
0100863
0100864
0100865
0100866
0100867
0100868
0100869
0100870
0100871
0100872
0100873
0100874
0100875
0100876
0100877
0100878
0100879
0100880
0100881
0100882
0100883
0100884
0100885
0100886
0100887
0100888
0100889
0100890
0100891
0100892
0100893
0100894
0100895
0100896
0100897
0100898
0100899
0100900
0100901
0100902
0100903
0100904
0100905
0100906
0100907
0100908
010090
```

```

00533         g->object.line.start.x *= -1.0;
00534         g->object.line.end.x *= -1.0;
00535     }
00536     if (vert) {
00537         g->object.line.start.y *= -1.0;
00538         g->object.line.end.y *= -1.0;
00539     }
00540     break;
00541 }
00542 case EMB_CIRCLE: {
00543     if (horz) {
00544         g->object.circle.center.x *= -1.0;
00545     }
00546     if (vert) {
00547         g->object.circle.center.y *= -1.0;
00548     }
00549     break;
00550 }
00551 case EMB_ELLIPSE: {
00552     if (horz) {
00553         g->object.ellipse.center.x *= -1.0;
00554     }
00555     if (vert) {
00556         g->object.ellipse.center.y *= -1.0;
00557     }
00558     break;
00559 }
00560 case EMB_PATH: {
00561     EmbArray *point_list = g->object.path.pointList;
00562     for (j=0; j < point_list->count; j++) {
00563         if (horz) {
00564             point_list->geometry[j].object.point.position.x *= -1.0;
00565         }
00566         if (vert) {
00567             point_list->geometry[j].object.point.position.y *= -1.0;
00568         }
00569     }
00570     break;
00571 }
00572 case EMB_POINT: {
00573     if (horz) {
00574         g->object.point.position.x *= -1.0;
00575     }
00576     if (vert) {
00577         g->object.point.position.y *= -1.0;
00578     }
00579     break;
00580 }
00581 case EMB_POLYGON: {
00582     EmbArray *point_list = g->object.polygon.pointList;
00583     for (j=0; j < point_list->count; j++) {
00584         if (horz) {
00585             point_list->geometry[i].object.point.position.x *= -1.0;
00586         }
00587         if (vert) {
00588             point_list->geometry[i].object.point.position.y *= -1.0;
00589         }
00590     }
00591 }
00592 case EMB_POLYLINE: {
00593     EmbArray *point_list = g->object.polygon.pointList;
00594     for (j=0; j < point_list->count; j++) {
00595         if (horz) {
00596             point_list->geometry[j].object.point.position.x *= -1.0;
00597         }
00598         if (vert) {
00599             point_list->geometry[j].object.point.position.y *= -1.0;
00600         }
00601     }
00602     break;
00603 }
00604 case EMB_RECT: {
00605     if (horz) {
00606         g->object.rect.left *= -1.0;
00607         g->object.rect.right *= -1.0;
00608     }
00609     if (vert) {
00610         g->object.rect.top *= -1.0;
00611         g->object.rect.bottom *= -1.0;
00612     }
00613     break;
00614 }
00615 case EMB_SPLINE: {
00616     /* TODO */
00617     break;
00618 default:
00619     break;
00620 }
```

```

00620      }
00621  }
00622
00623 void
00624 embPattern_combineJumpStitches(EmbPattern* p)
00625 {
00626     int jumpCount = 0, i;
00627     EmbArray *newList;
00628     EmbStitch j;
00629
00630     if (!p) {
00631         printf("ERROR: emb-pattern.c embPattern_combineJumpStitches(), ");
00632         printf("p argument is null\n");
00633         return;
00634     }
00635     newList = embArray_create(EMB_STITCH);
00636     for (i = 0; i < p->stitch_list->count; i++) {
00637         EmbStitch st = p->stitch_list->stitch[i];
00638         if (st.flags & JUMP) {
00639             if (jumpCount == 0) {
00640                 j = st;
00641             } else {
00642                 j.x += st.x;
00643                 j.y += st.y;
00644             }
00645             jumpCount++;
00646         } else {
00647             if (jumpCount > 0) {
00648                 embArray_addStitch(newList, j);
00649             }
00650             embArray_addStitch(newList, st);
00651         }
00652     }
00653     embArray_free(p->stitch_list);
00654     p->stitch_list = newList;
00655 }
00656
00657 /*TODO: The params determine the max XY movement rather than the length.
00658 They need renamed or clarified further. */
00659 void
00660 embPattern_correctForMaxStitchLength(EmbPattern* p,
00661                                     EmbReal maxStitchLength, EmbReal maxJumpLength)
00662 {
00663     if (!p) {
00664         printf("ERROR: emb-pattern.c embPattern_correctForMaxStitchLength(), ");
00665         printf("p argument is null\n");
00666         return;
00667     }
00668     if (p->stitch_list->count > 1) {
00669         int i, j, splits;
00670         EmbReal maxXY, maxLen, addX, addY;
00671         EmbArray *newList = embArray_create(EMB_STITCH);
00672         for (i=1; i < p->stitch_list->count; i++) {
00673             EmbStitch st = p->stitch_list->stitch[i];
00674             EmbReal xx = st.x;
00675             EmbReal yy = st.y;
00676             EmbReal dx = p->stitch_list->stitch[i-1].x - xx;
00677             EmbReal dy = p->stitch_list->stitch[i-1].y - yy;
00678             if ((fabs(dx) > maxStitchLength) || (fabs(dy) > maxStitchLength)) {
00679                 maxXY = EMB_MAX(fabs(dx), fabs(dy));
00680                 if (st.flags & (JUMP | TRIM)) {
00681                     maxLen = maxJumpLength;
00682                 } else {
00683                     maxLen = maxStitchLength;
00684                 }
00685                 splits = (int)ceil((EmbReal)maxXY / maxLen);
00686
00687                 if (splits > 1) {
00688                     addX = (EmbReal)dx / splits;
00689                     addY = (EmbReal)dy / splits;
00690
00691                     for (j = 1; j < splits; j++) {
00692                         EmbStitch s;
00693                         s = st;
00694                         s.x = xx + addX * j;
00695                         s.y = yy + addY * j;
00696                         embArray_addStitch(newList, s);
00697                     }
00698                 }
00699             }
00700             embArray_addStitch(newList, st);
00701         }
00702         embArray_free(p->stitch_list);
00703         p->stitch_list = newList;
00704     }
00705     embPattern_end(p);
00706 }
```

```
00707
00708 void
00709 embPattern_center(EmbPattern* p)
00710 {
00711     /* TODO: review this. currently not used in anywhere.
00712         Also needs to handle various design objects */
00713     int moveLeft, moveTop, i;
00714     EmbRect boundingRect;
00715     if (!p) {
00716         printf("ERROR: emb-pattern.c embPattern_center(), p argument is null\n");
00717         return;
00718     }
00719     boundingRect = embPattern_calcBoundingBox(p);
00720
00721     moveLeft = (int)(boundingRect.left - (boundingRect.right - boundingRect.left) / 2.0);
00722     moveTop = (int)(boundingRect.top - (boundingRect.bottom - boundingRect.top) / 2.0);
00723
00724     for (i = 0; i < p->stitch_list->count; i++) {
00725         p->stitch_list->stitch[i].x -= moveLeft;
00726         p->stitch_list->stitch[i].y -= moveTop;
00727     }
00728 }
00729
00730 /*TODO: Description needed. */
00731 void
00732 embPattern_loadExternalColorFile(EmbPattern* p, const char* fileName)
00733 {
00734     int hasRead, stub_len, format;
00735     char extractName[200];
00736
00737     if (!p) {
00738         printf("ERROR: emb-pattern.c embPattern_loadExternalColorFile(), p argument is null\n");
00739         return;
00740     }
00741     if (!fileName) {
00742         printf("ERROR: emb-pattern.c embPattern_loadExternalColorFile(), fileName argument is
00743             null\n");
00744         return;
00745     }
00746     strcpy(extractName, fileName);
00747     format = emb_identify_format(fileName);
00748     stub_len = strlen(fileName) - strlen(formatTable[format].extension);
00749     extractName[stub_len] = 0;
00750     strcat(extractName, ".edr");
00751     hasRead = embPattern_read(p, extractName, EMB_FORMAT_EDR);
00752     if (!hasRead) {
00753         extractName[stub_len] = 0;
00754         strcat(extractName, ".rgb");
00755         hasRead = embPattern_read(p, extractName, EMB_FORMAT_RGB);
00756     }
00757     if (!hasRead) {
00758         extractName[stub_len] = 0;
00759         strcat(extractName, ".col");
00760         hasRead = embPattern_read(p, extractName, EMB_FORMAT_COL);
00761     }
00762     if (!hasRead) {
00763         extractName[stub_len] = 0;
00764         strcat(extractName, ".inf");
00765         hasRead = embPattern_read(p, extractName, EMB_FORMAT_INF);
00766     }
00767 }
00768
00769 void
00770 embPattern_free(EmbPattern* p)
00771 {
00772     if (!p) {
00773         printf("ERROR: emb-pattern.c embPattern_free(), p argument is null\n");
00774         return;
00775     }
00776     embArray_free(p->stitch_list);
00777     embArray_free(p->thread_list);
00778     embArray_free(p->geometry);
00779     free(p);
00780 }
00781
00782
00783 void
00784 embPattern_addCircleAbs(EmbPattern* p, EmbCircle circle)
00785 {
00786     if (!p) {
00787         printf("ERROR: emb-pattern.c embPattern_addCircleObjectAbs(), p argument is null\n");
00788         return;
00789     }
00790     embArray_addCircle(p->geometry, circle);
00791 }
00792
00793
00794
00795 }
```

```
00800 void
00801 embPattern_addEllipseAbs(EmbPattern* p, EmbEllipse ellipse)
00802 {
00803     if (!p) {
00804         printf("ERROR: emb-pattern.c embPattern_addEllipseObjectAbs(), p argument is null\n");
00805         return;
00806     }
00807
00808     embArray_addEllipse(p->geometry, ellipse);
00809 }
00810
00815 void
00816 embPattern_addLineAbs(EmbPattern* p, EmbLine line)
00817 {
00818     if (!p) {
00819         printf("ERROR: emb-pattern.c embPattern_addLineObjectAbs(), p argument is null\n");
00820         return;
00821     }
00822
00823     embArray_addLine(p->geometry, line);
00824 }
00825
00826 void
00827 embPattern_addPathAbs(EmbPattern* p, EmbPath obj)
00828 {
00829     if (!p) {
00830         printf("ERROR: emb-pattern.c embPattern_addPathObjectAbs(), p argument is null\n");
00831         return;
00832     }
00833     if (!obj.pointList) {
00834         printf("ERROR: emb-pattern.c embPattern_addPathObjectAbs(), obj->pointList is empty\n");
00835         return;
00836     }
00837
00838     embArray_addPath(p->geometry, obj);
00839 }
00840
00842 void
00843 embPattern_addPointAbs(EmbPattern* p, EmbPoint obj)
00844 {
00845     if (!p) {
00846         printf("ERROR: emb-pattern.c embPattern_addPointObjectAbs(), p argument is null\n");
00847         return;
00848     }
00849
00850     embArray_addPoint(p->geometry, obj);
00851 }
00852
00853 void
00854 embPattern_addPolygonAbs(EmbPattern* p, EmbPolygon obj)
00855 {
00856     if (!p) {
00857         printf("ERROR: emb-pattern.c embPattern_addPolygonObjectAbs(), p argument is null\n");
00858         return;
00859     }
00860     if (!obj.pointList) {
00861         printf("ERROR: emb-pattern.c embPattern_addPolygonObjectAbs(), obj->pointList is empty\n");
00862         return;
00863     }
00864
00865     embArray_addPolygon(p->geometry, obj);
00866 }
00867
00868 void
00869 embPattern_addPolylineObjectAbs(EmbPattern* p, EmbPolyline obj)
00870 {
00871     if (!p) {
00872         printf("ERROR: emb-pattern.c embPattern_addPolylineObjectAbs(), p argument is null\n");
00873         return;
00874     }
00875     if (!obj.pointList) {
00876         printf("ERROR: emb-pattern.c embPattern_addPolylineObjectAbs(), obj->pointList is empty\n");
00877         return;
00878     }
00879     embArray_addPolyline(p->geometry, obj);
00880 }
00881
00887 void
00888 embPattern_addRectAbs(EmbPattern* p, EmbRect rect)
00889 {
00890     if (!p) {
00891         printf("ERROR: emb-pattern.c embPattern_addRectObjectAbs(), p argument is null\n");
00892         return;
00893     }
00894     embArray_addRect(p->geometry, rect);
00895 }
```

```

00897 void
00898 embPattern_end(EmbPattern *p)
00899 {
00900     if (p->stitch_list->count == 0) {
00901         return;
00902     }
00903     /* Check for an END stitch and add one if it is not present */
00904     if (p->stitch_list->stitch[p->stitch_list->count-1].flags != END) {
00905         embPattern_addStitchRel(p, 0, 0, END, 1);
00906     }
00907 }
00908
00909
00910 int
00911 embPattern_color_count(EmbPattern *pattern, EmbColor startColor)
00912 {
00913     int number_of_colors = 0, i;
00914     EmbColor color = startColor;
00915     for (i=0; i<pattern->stitch_list->count; i++) {
00916         EmbColor newColor;
00917         EmbStitch st;
00918
00919         st = pattern->stitch_list->stitch[i];
00920
00921         newColor = pattern->thread_list->thread[st.color].color;
00922         if (embColor_distance(newColor, color) != 0) {
00923             number_of_colors++;
00924             color = newColor;
00925         }
00926         else if (st.flags & END || st.flags & STOP) {
00927             number_of_colors++;
00928         }
00929
00930         while (pattern->stitch_list->stitch[i+1].flags == st.flags) {
00931             i++;
00932             if (i >= pattern->stitch_list->count-2) {
00933                 break;
00934             }
00935         }
00936     }
00937     return number_of_colors;
00938 }
00939
00940
00941 void
00942 embPattern_designDetails(EmbPattern *pattern)
00943 {
00944     int colors, num_stitches, real_stitches, jump_stitches, trim_stitches;
00945     int unknown_stitches;
00946     EmbRect bounds;
00947
00948     puts("Design Details");
00949     bounds = embPattern_calcBoundingBox(pattern);
00950
00951     colors = 1;
00952     num_stitches = pattern->stitch_list->count;
00953     real_stitches = 0;
00954     jump_stitches = 0;
00955     trim_stitches = 0;
00956     unknown_stitches = 0;
00957     bounds = embPattern_calcBoundingBox(pattern);
00958
00959     if (emb_verbose > 1) {
00960         printf("colors: %d\n", colors);
00961         printf("num_stitches: %d\n", num_stitches);
00962         printf("real_stitches: %d\n", real_stitches);
00963         printf("jump_stitches: %d\n", jump_stitches);
00964         printf("trim_stitches: %d\n", trim_stitches);
00965         printf("unknown_stitches: %d\n", unknown_stitches);
00966         printf("num_colors: %d\n", pattern->thread_list->count);
00967         printf("bounds.left: %f\n", bounds.left);
00968         printf("bounds.right: %f\n", bounds.right);
00969         printf("bounds.top: %f\n", bounds.top);
00970         printf("bounds.bottom: %f\n", bounds.bottom);
00971     }
00972 /*
00973     EmbReal minx = 0.0, maxx = 0.0, miny = 0.0, maxy = 0.0;
00974     EmbReal min_stitchlength = 999.0;
00975     EmbReal max_stitchlength = 0.0;
00976     EmbReal total_stitchlength = 0.0;
00977     int number_of_minlength_stitches = 0;
00978     int number_of_maxlength_stitches = 0;
00979
00980     EmbReal xx = 0.0, yy = 0.0;
00981     EmbReal length = 0.0;
00982
00983     if (num_stitches == 0) {

```

```

00984     QMessageBox::warning(this, tr("No Design Loaded"), tr("<b>A design needs to be loaded or  
00985     created before details can be determined.</b>"));
00986 }
00987 QVector<EmbReal> stitchLengths;
00988
00989 EmbReal totalColorLength = 0.0;
00990 int i;
00991 for (i = 0; i < num_stitches; i++) {
00992     EmbStitch st = pattern->stitch_list->stitch[i];
00993     EmbReal dx, dy;
00994     dx = st.x - xx;
00995     dy = st.y - yy;
00996     xx = st.x;
00997     yy = st.y;
00998     length=sqrt(dx * dx + dy * dy);
00999     totalColorLength += length;
01000     if(i > 0 && embstitch_list_getAt(pattern->stitch_list, i-1).flags != NORMAL)
01001         length = 0.0; //can't count first normal stitch;
01002     if(!(embstitch_list_getAt(pattern->stitch_list, i).flags & (JUMP | TRIM)))
01003     {
01004         real_stitches++;
01005         if(length > max_stitchlength) { max_stitchlength = length; number_of_maxlength_stitches =
0; }
01006         if(length == max_stitchlength) number_of_maxlength_stitches++;
01007         if(length > 0 && length < min_stitchlength)
01008         {
01009             min_stitchlength = length;
01010             number_of_minlength_stitches = 0;
01011         }
01012         if(length == min_stitchlength) number_of_minlength_stitches++;
01013         total_stitchlength += length;
01014         if(xx < minx) minx = xx;
01015         if(xx > maxx) maxx = xx;
01016         if(yy < miny) miny = yy;
01017         if(yy > maxy) maxy = yy;
01018     }
01019     if (st.flags & JUMP) {
01020         jump_stitches++;
01021     }
01022     if (st.flags & TRIM) {
01023         trim_stitches++;
01024     }
01025     if (st.flags & STOP) {
01026         stitchLengths.push_back(totalColorLength);
01027         totalColorLength = 0;
01028         colors++;
01029     }
01030     if (st.flags & END) {
01031         stitchLengths.push_back(totalColorLength);
01032     }
01033 }
01034
01035 //second pass to fill bins now that we know max stitch length
01036 #define NUMBINS 10
01037 int bin[NUMBINS+1];
01038 int i;
01039 for (i = 0; i <= NUMBINS; i++) {
01040     bin[i]=0;
01041 }
01042
01043 for (i = 0; i < num_stitches; i++) {
01044     dx = embstitch_list_getAt(pattern->stitch_list, i).xx - xx;
01045     dy = embstitch_list_getAt(pattern->stitch_list, i).yy - yy;
01046     xx = embstitch_list_getAt(pattern->stitch_list, i).xx;
01047     yy = embstitch_list_getAt(pattern->stitch_list, i).yy;
01048     if(i > 0 && embstitch_list_getAt(pattern->stitch_list, i-1).flags == NORMAL &&
01049        embstitch_list_getAt(pattern->stitch_list, i).flags == NORMAL)
01050     {
01051         length=sqrt(dx * dx + dy * dy);
01052         bin[int(floor(NUMBINS*length/max_stitchlength))]++;
01053     }
01054
01055 EmbReal binSize = max_stitchlength / NUMBINS;
01056
01057 QString str;
01058 int i;
01059 for (i = 0; i < NUMBINS; i++) {
01060     str += QString::number(binSize * (i), 'f', 1) + " - " + QString::number(binSize * (i+1), 'f',
01061     1) + " mm: " + QString::number(bin[i]) + "\n\n";
01062 }
01063 puts("Stitches: %d\n", num_stitches);
01064 puts("Colors: %d\n", num_colors);
01065 puts("Jumps: %d\n", jump_stitches);
01066 puts("Top: %f mm", bounds.top);

```

```

01067     puts("Left: %f mm", bounds.left);
01068     puts("Bottom: %f mm", bounds.bottom);
01069     puts("Right: %f mm", bounds.right);
01070     puts("Width: %f mm", bounds.right - bounds.left);
01071     puts("Height: %f mm", bounds.bottom - bounds.top);
01072     grid->addWidget(new QLabel(tr("\nStitch Distribution: \n")), 9, 0, 1, 2);
01073     grid->addWidget(new QLabel(str), 10, 0, 1, 1);
01074     grid->addWidget(new QLabel(tr("\nThread Length By Color: \n")), 11, 0, 1, 2);
01075     int currentRow = 12;
01076
01077     int i;
01078     for (i = 0; i < num_colors; i++) {
01079         QFrame *frame = new QFrame();
01080         frame->setGeometry(0, 0, 30, 30);
01081         QPalette palette = frame->palette();
01082         EmbColor t = embThreadList_getAt(pattern->threadList, i).color;
01083         palette.setColor(backgroundRole(), QColor(t.r, t.g, t.b));
01084         frame->setPalette(palette);
01085         frame->setAutoFillBackground(true);
01086         grid->addWidget(frame, currentRow, 0, 1, 1);
01087         debug_message("size: %d i: %d", stitchLengths.size(), i);
01088         grid->addWidget(new QLabel(QString::number(stitchLengths.at(i)) + " mm"), currentRow, 1, 1, 1);
01089         currentRow++;
01090     }
01091
01092     QDialogButtonBox buttonbox(Qt::Horizontal, &dialog);
01093     QPushButton button(&dialog);
01094     button.setText("Ok");
01095     buttonbox.addButton(&button, QDialogButtonBox::AcceptRole);
01096     buttonbox.setCenterButtons(true);
01097     connect(&buttonbox, SIGNAL(accepted()), &dialog, SLOT(accept()));
01098
01099     grid->addWidget(&buttonbox, currentRow, 0, 1, 2);
01100 */
01101 }
01102
01103
01104 /*
01105 *
01106 */
01107 int
01108 convert(const char *inf, const char *outf)
01109 {
01110     EmbPattern* p = 0;
01111     int reader, writer;
01112
01113     p = embPattern_create();
01114     if (!p) {
01115         printf("ERROR: convert(), cannot allocate memory for p\n");
01116         return 1;
01117     }
01118
01119     if (!embPattern_readAuto(p, inf)) {
01120         printf("ERROR: convert(), reading file was unsuccessful: %s\n", inf);
01121         embPattern_free(p);
01122         return 1;
01123     }
01124
01125     reader = emb_identify_format(inf);
01126     writer = emb_identify_format(outf);
01127     if (formatTable[reader].type == EMBFORMAT_OBJECTONLY) {
01128         if (formatTable[writer].type == EMBFORMAT_STITCHONLY) {
01129             embPattern_movePolylinesToStitch_list(p);
01130         }
01131     }
01132
01133     if (!embPattern_writeAuto(p, outf)) {
01134         printf("ERROR: convert(), writing file %s was unsuccessful\n", outf);
01135         embPattern_free(p);
01136         return 1;
01137     }
01138
01139     embPattern_free(p);
01140     return 0;
01141 }
01142
01143 float embPattern_totalStitchLength(EmbPattern *pattern)
01144 {
01145     EmbArray *sts = pattern->stitch_list;
01146     float result = 0.0;
01147     int i;
01148     for (i = 1; i < sts->count; i++) {
01149         EmbStitch st = sts->stitch[i];
01150         EmbReal length = 0.0;
01151         EmbVector delta;
01152         delta.x = st.x - sts->stitch[i-1].x;
01153         delta.y = st.y - sts->stitch[i-1].y;

```

```

01154     length = embVector_length(delta);
01155     if (st.flags & NORMAL)
01156         if (sts->stitch[i-1].flags & NORMAL) {
01157             result += length;
01158         }
01159     }
01160     return result;
01161 }
01162
01163 float embPattern_minimumStitchLength(EmbPattern *pattern)
01164 {
01165     EmbArray *sts = pattern->stitch_list;
01166     float result = 1.0e10;
01167     int i;
01168     for (i = 1; i < sts->count; i++) {
01169         EmbReal length = 0.0;
01170         EmbVector delta;
01171         delta.x = sts->stitch[i].x - sts->stitch[i-1].x;
01172         delta.y = sts->stitch[i].y - sts->stitch[i-1].y;
01173         length = embVector_length(delta);
01174         if (sts->stitch[i].flags & NORMAL)
01175             if (sts->stitch[i-1].flags & NORMAL) {
01176                 if (length < result) {
01177                     result = length;
01178                 }
01179             }
01180     }
01181     return result;
01182 }
01183
01184 float embPattern_maximumStitchLength(EmbPattern *pattern)
01185 {
01186     EmbArray *sts = pattern->stitch_list;
01187     float result = 0.0;
01188     int i;
01189     for (i = 1; i < sts->count; i++) {
01190         EmbReal length = 0.0;
01191         EmbVector delta;
01192         delta.x = sts->stitch[i].x - sts->stitch[i-1].x;
01193         delta.y = sts->stitch[i].y - sts->stitch[i-1].y;
01194         length = embVector_length(delta);
01195         if (sts->stitch[i].flags & NORMAL)
01196             if (sts->stitch[i-1].flags & NORMAL) {
01197                 if (length > result) {
01198                     result = length;
01199                 }
01200             }
01201     }
01202     return result;
01203 }
01204
01205 void embPattern_lengthHistogram(EmbPattern *pattern, int *bin, int NUMBINS)
01206 {
01207     int i;
01208     float max_stitch_length = embPattern_maximumStitchLength(pattern);
01209     EmbArray *sts = pattern->stitch_list;
01210     for (i = 0; i <= NUMBINS; i++) {
01211         bin[i] = 0;
01212     }
01213
01214     for (i = 1; i < sts->count; i++) {
01215         if (sts->stitch[i].flags & NORMAL)
01216             if (sts->stitch[i-1].flags & NORMAL) {
01217                 EmbVector delta;
01218                 float length;
01219                 delta.x = sts->stitch[i].x - sts->stitch[i-1].x;
01220                 delta.y = sts->stitch[i].y - sts->stitch[i-1].y;
01221                 length = embVector_length(delta);
01222                 bin[(int)(floor(NUMBINS*length/max_stitch_length))]++;
01223             }
01224     }
01225 }
01226
01227 int embPattern_realStitches(EmbPattern *pattern)
01228 {
01229     int i;
01230     EmbArray *sts = pattern->stitch_list;
01231     int real_stitches = 0;
01232     for (i = 0; i < sts->count; i++) {
01233         if (!(sts->stitch[i].flags & (JUMP | TRIM | END))) {
01234             real_stitches++;
01235         }
01236     }
01237     return real_stitches;
01238 }
01239
01240 int embPattern_jumpStitches(EmbPattern *pattern)

```

```

01241 {
01242     int i;
01243     EmbArray *sts = pattern->stitch_list;
01244     int jump_stitches = 0;
01245     for (i = 0; i < sts->count; i++) {
01246         if (sts->stitch[i].flags & JUMP) {
01247             jump_stitches++;
01248         }
01249     }
01250     return jump_stitches;
01251 }
01252
01253
01254 int embPattern_trimStitches(EmbPattern *pattern)
01255 {
01256     int i;
01257     EmbArray *sts = pattern->stitch_list;
01258     int trim_stitches = 0;
01259     for (i = 0; i < sts->count; i++) {
01260         if (sts->stitch[i].flags & TRIM) {
01261             trim_stitches++;
01262         }
01263     }
01264     return trim_stitches;
01265 }
```

4.23 src/thread-color.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "embroidery_internal.h"
```

Functions

- int **threadColor** (const char *name, int brand)
- int **threadColorNum** (unsigned int color, int brand)
- const char * **threadColorName** (unsigned int color, int brand)

Variables

- const unsigned char **_dxfColorTable** [][][3] = {{ { 0, 0, 0 } }}
- const **EmbThread** **husThreads** [] = {{{ 0, 0, 0 }, "END", "END" }}
- const **EmbThread** **jefThreads** [] = {{{ 0, 0, 0 }, "END", "END" }}
- const **EmbThread** **shvThreads** [] = {{{ 0, 0, 0 }, "END", "END" }}
- const **EmbThread** **pcmThreads** [] = {{{ 0, 0, 0 }, "END", "END" }}
- const **EmbThread** **pecThreads** [] = {{{ 0, 0, 0 }, "END", "END" }}
- const int **shvThreadCount** = 42
- const int **pecThreadCount** = 65
- **thread_color** * **brand_codes** []
- const char * **brand_codes_files** []

4.23.1 Function Documentation

4.23.1.1 `threadColor()`

```
int threadColor (
    const char * name,
    int brand )
```

Definition at line [4012](#) of file [thread-color.c](#).

4.23.1.2 `threadColorName()`

```
const char * threadColorName (
    unsigned int color,
    int brand )
```

Definition at line [4035](#) of file [thread-color.c](#).

4.23.1.3 `threadColorNum()`

```
int threadColorNum (
    unsigned int color,
    int brand )
```

Definition at line [4023](#) of file [thread-color.c](#).

4.23.2 Variable Documentation

4.23.2.1 `_dxfColorTable`

```
const unsigned char _dxfColorTable[ ][3] = {{ 0, 0, 0 }}
```

Definition at line [14](#) of file [thread-color.c](#).

4.23.2.2 `brand_codes`

```
thread\_color* brand_codes[ ]
```

Definition at line [23](#) of file [thread-color.c](#).

4.23.2.3 brand_codes_files

```
const char* brand_codes_files[ ]
```

Initial value:

```
= {
    "arc_polyester_colors.csv",
    "arc_rayon_colors.csv",
    "coats_and_clark_rayon_colors.csv",
    "exquisite_polyester_colors.csv",
    "fufu_Polyester_colors.csv",
    "fufu_Rayon_colors.csv",
    "Hemingworth_Polyester_colors.csv",
    "Isacord_Polyester_colors.csv",
    "Isafil_Rayon_colors.csv",
    "Marathon_Polyester_colors.csv",
    "Marathon_Rayon_colors.csv",
    "Madeira_Polyester_colors.csv",
    "Madeira_Rayon_colors.csv",
    "Metro_Polyester_colors.csv",
    "Pantone_colors.csv",
    "RobisonAnton_Polyester_colors.csv",
    "RobisonAnton_Rayon_colors.csv",
    "Sigma_Polyester_colors.csv",
    "Sulky_Rayon_colors.csv",
    "ThreadArt_Rayon_colors.csv",
    "ThreadArt_Polyester_colors.csv",
    "ThreadDelight_Polyester_colors.csv",
    "Z102_Isacord_Polyester_colors.csv",
    "svg_color_colors.csv"
}
```

Definition at line 58 of file [thread-color.c](#).

4.23.2.4 husThreads

```
const EmbThread husThreads[ ] = {{{ 0, 0, 0 }, "END", "END" }};
```

Definition at line 15 of file [thread-color.c](#).

4.23.2.5 jefThreads

```
const EmbThread jefThreads[ ] = {{{ 0, 0, 0 }, "END", "END" }};
```

Definition at line 16 of file [thread-color.c](#).

4.23.2.6 pcmThreads

```
const EmbThread pcmThreads[ ] = {{{ 0, 0, 0 }, "END", "END" }};
```

Definition at line 18 of file [thread-color.c](#).

4.23.2.7 pecThreadCount

```
const int pecThreadCount = 65
```

Definition at line 21 of file [thread-color.c](#).

4.23.2.8 pecThreads

```
const EmbThread pecThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
```

Definition at line 19 of file [thread-color.c](#).

4.23.2.9 shvThreadCount

```
const int shvThreadCount = 42
```

Definition at line 20 of file [thread-color.c](#).

4.23.2.10 shvThreads

```
const EmbThread shvThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
```

Definition at line 17 of file [thread-color.c](#).

4.24 thread-color.c

[Go to the documentation of this file.](#)

```
00001 /*
00002  * This file is part of libembroidery.
00003  *
00004  * Copyright 2018-2022 The Embroidermodder Team
00005  * Licensed under the terms of the zlib license.
00006 */
00007
00008 #include <stdio.h>
00009 #include <string.h>
00010
00011 #include "embroidery_internal.h"
00012
00013 #ifndef LIBEMBROIDERY_CLI
00014 const unsigned char _dxfColorTable[][3] = {{ 0, 0, 0 }};
00015 const EmbThread husThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
00016 const EmbThread jefThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
00017 const EmbThread shvThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
00018 const EmbThread pcmThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
00019 const EmbThread pecThreads[] = {{{ 0, 0, 0 }, "END", "END" }};
00020 const int shvThreadCount = 42;
00021 const int pecThreadCount = 65;
00022
00023 thread_color *brand_codes[] = {
00024     NULL,
00025     NULL,
00026     NULL,
```

```
00027     NULL,
00028     NULL,
00029     NULL,
00030     NULL,
00031     NULL,
00032     NULL,
00033     NULL,
00034     NULL,
00035     NULL,
00036     NULL,
00037     NULL,
00038     NULL,
00039     NULL,
00040     NULL,
00041     NULL,
00042     NULL,
00043     NULL,
00044     NULL,
00045     NULL,
00046     NULL,
00047     NULL,
00048     NULL,
00049     NULL,
00050     NULL,
00051     NULL,
00052     NULL,
00053     NULL,
00054     NULL,
00055     NULL,
00056 };
00057
00058 const char *brand_codes_files[] = {
00059     "arc_polyester_colors.csv", /* 0 */
00060     "arc_rayon_colors.csv", /* 1 */
00061     "coats_and_clark_rayon_colors.csv", /* 2 */
00062     "exquisite_polyester_colors.csv", /* 3 */
00063     "fufu_Polyester_colors.csv", /* 4 */
00064     "fufu_Rayon_colors.csv", /* 5 */
00065     "Hemingworth_Polyester_colors.csv", /* 6 */
00066     "Isacord_Polyester_colors.csv", /* 7 */
00067     "Isafil_Rayon_colors.csv", /* 8 */
00068     "Marathon_Polyester_colors.csv", /* 9 */
00069     "Marathon_Rayon_colors.csv", /* 10 */
00070     "Madeira_Polyester_colors.csv", /* 11 */
00071     "Madeira_Rayon_colors.csv", /* 12 */
00072     "Metro_Polyester_colors.csv", /* 13 */
00073     "Pantone_colors.csv", /* 14 */
00074     "RobisonAnton_Polyester_colors.csv", /* 15 */
00075     "RobisonAnton_Rayon_colors.csv", /* 16 */
00076     "Sigma_Polyester_colors.csv", /* 17 */
00077     "Sulky_Rayon_colors.csv", /* 18 */
00078     "ThreadArt_Rayon_colors.csv", /* 19 */
00079     "ThreadArt_Polyester_colors.csv", /* 20 */
00080     "ThreaDelight_Polyester_colors.csv", /* 21 */
00081     "Z102_Isacord_Polyester_colors.csv", /* 22 */
00082     "svg_color_colors.csv" /* 23 */
00083 };
00084 #else
00085
00086
00087 /* Based on the DraftSight color table */
00088 const unsigned char _dxfColorTable[][][3] = {
00089 { 0, 0, 0 }, /* '0' (BYBLOCK) */
00090 { 255, 0, 0 }, /* '1' (red) */
00091 { 255, 255, 0 }, /* '2' (yellow) */
00092 { 0, 255, 0 }, /* '3' (green) */
00093 { 0, 255, 255 }, /* '4' (cyan) */
00094 { 0, 0, 255 }, /* '5' (blue) */
00095 { 255, 0, 255 }, /* '6' (magenta) */
00096 { 255, 255, 255 }, /* '7' (white) */
00097 { 128, 128, 128 }, /* '8' (dark gray) */
00098 { 192, 192, 192 }, /* '9' (light gray) */
00099 { 255, 0, 0 }, /* '10' */
00100 { 255, 127, 127 }, /* '11' */
00101 { 204, 0, 0 }, /* '12' */
00102 { 204, 102, 102 }, /* '13' */
00103 { 153, 0, 0 }, /* '14' */
00104 { 153, 76, 76 }, /* '15' */
00105 { 127, 0, 0 }, /* '16' */
00106 { 127, 63, 63 }, /* '17' */
00107 { 76, 0, 0 }, /* '18' */
00108 { 76, 38, 38 }, /* '19' */
00109 { 255, 63, 0 }, /* '20' */
00110 { 255, 159, 127 }, /* '21' */
00111 { 204, 51, 0 }, /* '22' */
00112 { 204, 127, 102 }, /* '23' */
00113 { 153, 38, 0 }, /* '24' */
```

```
00114 { 153, 95, 76 }, /* '25' */
00115 { 127, 31, 0 }, /* '26' */
00116 { 127, 79, 63 }, /* '27' */
00117 { 76, 19, 0 }, /* '28' */
00118 { 76, 47, 38 }, /* '29' */
00119 { 255, 127, 0 }, /* '30' */
00120 { 255, 191, 127 }, /* '31' */
00121 { 204, 102, 0 }, /* '32' */
00122 { 204, 153, 102 }, /* '33' */
00123 { 153, 76, 0 }, /* '34' */
00124 { 153, 114, 76 }, /* '35' */
00125 { 127, 63, 0 }, /* '36' */
00126 { 127, 95, 63 }, /* '37' */
00127 { 76, 38, 0 }, /* '38' */
00128 { 76, 57, 38 }, /* '39' */
00129 { 255, 191, 0 }, /* '40' */
00130 { 255, 223, 127 }, /* '41' */
00131 { 204, 153, 0 }, /* '42' */
00132 { 204, 178, 102 }, /* '43' */
00133 { 153, 114, 0 }, /* '44' */
00134 { 153, 133, 76 }, /* '45' */
00135 { 127, 95, 0 }, /* '46' */
00136 { 127, 111, 63 }, /* '47' */
00137 { 76, 57, 0 }, /* '48' */
00138 { 76, 66, 38 }, /* '49' */
00139 { 255, 255, 0 }, /* '50' */
00140 { 255, 255, 127 }, /* '51' */
00141 { 204, 204, 0 }, /* '52' */
00142 { 204, 204, 102 }, /* '53' */
00143 { 153, 153, 0 }, /* '54' */
00144 { 153, 153, 76 }, /* '55' */
00145 { 127, 127, 0 }, /* '56' */
00146 { 127, 127, 63 }, /* '57' */
00147 { 76, 76, 0 }, /* '58' */
00148 { 76, 76, 38 }, /* '59' */
00149 { 191, 255, 0 }, /* '60' */
00150 { 223, 255, 127 }, /* '61' */
00151 { 153, 204, 0 }, /* '62' */
00152 { 178, 204, 102 }, /* '63' */
00153 { 114, 153, 0 }, /* '64' */
00154 { 133, 153, 76 }, /* '65' */
00155 { 95, 127, 0 }, /* '66' */
00156 { 111, 127, 63 }, /* '67' */
00157 { 57, 76, 0 }, /* '68' */
00158 { 66, 76, 38 }, /* '69' */
00159 { 127, 255, 0 }, /* '70' */
00160 { 191, 255, 127 }, /* '71' */
00161 { 102, 204, 0 }, /* '72' */
00162 { 153, 204, 102 }, /* '73' */
00163 { 76, 153, 0 }, /* '74' */
00164 { 114, 153, 76 }, /* '75' */
00165 { 63, 127, 0 }, /* '76' */
00166 { 95, 127, 63 }, /* '77' */
00167 { 38, 76, 0 }, /* '78' */
00168 { 57, 76, 38 }, /* '79' */
00169 { 63, 255, 0 }, /* '80' */
00170 { 159, 255, 127 }, /* '81' */
00171 { 51, 204, 0 }, /* '82' */
00172 { 127, 204, 102 }, /* '83' */
00173 { 38, 153, 0 }, /* '84' */
00174 { 95, 153, 76 }, /* '85' */
00175 { 31, 127, 0 }, /* '86' */
00176 { 79, 127, 63 }, /* '87' */
00177 { 19, 76, 0 }, /* '88' */
00178 { 47, 76, 38 }, /* '89' */
00179 { 0, 255, 0 }, /* '90' */
00180 { 127, 255, 127 }, /* '91' */
00181 { 0, 204, 0 }, /* '92' */
00182 { 102, 204, 102 }, /* '93' */
00183 { 0, 153, 0 }, /* '94' */
00184 { 76, 153, 76 }, /* '95' */
00185 { 0, 127, 0 }, /* '96' */
00186 { 63, 127, 63 }, /* '97' */
00187 { 0, 76, 0 }, /* '98' */
00188 { 38, 76, 38 }, /* '99' */
00189 { 0, 255, 63 }, /* '100' */
00190 { 127, 255, 159 }, /* '101' */
00191 { 0, 204, 51 }, /* '102' */
00192 { 102, 204, 127 }, /* '103' */
00193 { 0, 153, 38 }, /* '104' */
00194 { 76, 153, 95 }, /* '105' */
00195 { 0, 127, 31 }, /* '106' */
00196 { 63, 127, 79 }, /* '107' */
00197 { 0, 76, 19 }, /* '108' */
00198 { 38, 76, 47 }, /* '109' */
00199 { 0, 255, 127 }, /* '110' */
00200 { 127, 255, 191 }, /* '111' */
```

```
00201 { 0, 204, 102 }, /* '112' */
00202 { 102, 204, 153 }, /* '113' */
00203 { 0, 153, 76 }, /* '114' */
00204 { 76, 153, 114 }, /* '115' */
00205 { 0, 127, 63 }, /* '116' */
00206 { 63, 127, 95 }, /* '117' */
00207 { 0, 76, 38 }, /* '118' */
00208 { 38, 76, 57 }, /* '119' */
00209 { 0, 255, 191 }, /* '120' */
00210 { 127, 255, 223 }, /* '121' */
00211 { 0, 204, 153 }, /* '122' */
00212 { 102, 204, 178 }, /* '123' */
00213 { 0, 153, 114 }, /* '124' */
00214 { 76, 153, 133 }, /* '125' */
00215 { 0, 127, 95 }, /* '126' */
00216 { 63, 127, 111 }, /* '127' */
00217 { 0, 76, 57 }, /* '128' */
00218 { 38, 76, 66 }, /* '129' */
00219 { 0, 255, 255 }, /* '130' */
00220 { 127, 255, 255 }, /* '131' */
00221 { 0, 204, 204 }, /* '132' */
00222 { 102, 204, 204 }, /* '133' */
00223 { 0, 153, 153 }, /* '134' */
00224 { 76, 153, 153 }, /* '135' */
00225 { 0, 127, 127 }, /* '136' */
00226 { 63, 127, 127 }, /* '137' */
00227 { 0, 76, 76 }, /* '138' */
00228 { 38, 76, 76 }, /* '139' */
00229 { 0, 191, 255 }, /* '140' */
00230 { 127, 223, 255 }, /* '141' */
00231 { 0, 153, 204 }, /* '142' */
00232 { 102, 178, 204 }, /* '143' */
00233 { 0, 114, 153 }, /* '144' */
00234 { 76, 133, 153 }, /* '145' */
00235 { 0, 95, 127 }, /* '146' */
00236 { 63, 111, 127 }, /* '147' */
00237 { 0, 57, 76 }, /* '148' */
00238 { 38, 66, 76 }, /* '149' */
00239 { 0, 127, 255 }, /* '150' */
00240 { 127, 191, 255 }, /* '151' */
00241 { 0, 102, 204 }, /* '152' */
00242 { 102, 153, 204 }, /* '153' */
00243 { 0, 76, 153 }, /* '154' */
00244 { 76, 114, 153 }, /* '155' */
00245 { 0, 63, 127 }, /* '156' */
00246 { 63, 95, 127 }, /* '157' */
00247 { 0, 38, 76 }, /* '158' */
00248 { 38, 57, 76 }, /* '159' */
00249 { 0, 63, 255 }, /* '160' */
00250 { 127, 159, 255 }, /* '161' */
00251 { 0, 51, 204 }, /* '162' */
00252 { 102, 127, 204 }, /* '163' */
00253 { 0, 38, 153 }, /* '164' */
00254 { 76, 95, 153 }, /* '165' */
00255 { 0, 31, 127 }, /* '166' */
00256 { 63, 79, 127 }, /* '167' */
00257 { 0, 19, 76 }, /* '168' */
00258 { 38, 47, 76 }, /* '169' */
00259 { 0, 0, 255 }, /* '170' */
00260 { 127, 127, 255 }, /* '171' */
00261 { 0, 0, 204 }, /* '172' */
00262 { 102, 102, 204 }, /* '173' */
00263 { 0, 0, 153 }, /* '174' */
00264 { 76, 76, 153 }, /* '175' */
00265 { 0, 0, 127 }, /* '176' */
00266 { 63, 63, 127 }, /* '177' */
00267 { 0, 0, 76 }, /* '178' */
00268 { 38, 38, 76 }, /* '179' */
00269 { 63, 0, 255 }, /* '180' */
00270 { 159, 127, 255 }, /* '181' */
00271 { 51, 0, 204 }, /* '182' */
00272 { 127, 102, 204 }, /* '183' */
00273 { 38, 0, 153 }, /* '184' */
00274 { 95, 76, 153 }, /* '185' */
00275 { 31, 0, 127 }, /* '186' */
00276 { 79, 63, 127 }, /* '187' */
00277 { 19, 0, 76 }, /* '188' */
00278 { 47, 38, 76 }, /* '189' */
00279 { 127, 0, 255 }, /* '190' */
00280 { 191, 127, 255 }, /* '191' */
00281 { 102, 0, 204 }, /* '192' */
00282 { 153, 102, 204 }, /* '193' */
00283 { 76, 0, 153 }, /* '194' */
00284 { 114, 76, 153 }, /* '195' */
00285 { 63, 0, 127 }, /* '196' */
00286 { 95, 63, 127 }, /* '197' */
00287 { 38, 0, 76 }, /* '198' */
```

```

00288 { 57, 38, 76 }, /* '199' */
00289 { 191, 0, 255 }, /* '200' */
00290 { 223, 127, 255 }, /* '201' */
00291 { 153, 0, 204 }, /* '202' */
00292 { 178, 102, 204 }, /* '203' */
00293 { 114, 0, 153 }, /* '204' */
00294 { 133, 76, 153 }, /* '205' */
00295 { 95, 0, 127 }, /* '206' */
00296 { 111, 63, 127 }, /* '207' */
00297 { 57, 0, 76 }, /* '208' */
00298 { 66, 38, 76 }, /* '209' */
00299 { 255, 0, 255 }, /* '210' */
00300 { 255, 127, 255 }, /* '211' */
00301 { 204, 0, 204 }, /* '212' */
00302 { 204, 102, 204 }, /* '213' */
00303 { 153, 0, 153 }, /* '214' */
00304 { 153, 76, 153 }, /* '215' */
00305 { 127, 0, 127 }, /* '216' */
00306 { 127, 63, 127 }, /* '217' */
00307 { 76, 0, 76 }, /* '218' */
00308 { 76, 38, 76 }, /* '219' */
00309 { 255, 0, 191 }, /* '220' */
00310 { 255, 127, 223 }, /* '221' */
00311 { 204, 0, 153 }, /* '222' */
00312 { 204, 102, 178 }, /* '223' */
00313 { 153, 0, 114 }, /* '224' */
00314 { 153, 76, 133 }, /* '225' */
00315 { 127, 0, 95 }, /* '226' */
00316 { 127, 63, 111 }, /* '227' */
00317 { 76, 0, 57 }, /* '228' */
00318 { 76, 38, 66 }, /* '229' */
00319 { 255, 0, 127 }, /* '230' */
00320 { 255, 127, 191 }, /* '231' */
00321 { 204, 0, 102 }, /* '232' */
00322 { 204, 102, 153 }, /* '233' */
00323 { 153, 0, 76 }, /* '234' */
00324 { 153, 76, 114 }, /* '235' */
00325 { 127, 0, 63 }, /* '236' */
00326 { 127, 63, 95 }, /* '237' */
00327 { 76, 0, 38 }, /* '238' */
00328 { 76, 38, 57 }, /* '239' */
00329 { 255, 0, 63 }, /* '240' */
00330 { 255, 127, 159 }, /* '241' */
00331 { 204, 0, 51 }, /* '242' */
00332 { 204, 102, 127 }, /* '243' */
00333 { 153, 0, 38 }, /* '244' */
00334 { 153, 76, 95 }, /* '245' */
00335 { 127, 0, 31 }, /* '246' */
00336 { 127, 63, 79 }, /* '247' */
00337 { 76, 0, 19 }, /* '248' */
00338 { 76, 38, 47 }, /* '249' */
00339 { 51, 51, 51 }, /* '250' */
00340 { 91, 91, 91 }, /* '251' */
00341 { 132, 132, 132 }, /* '252' */
00342 { 173, 173, 173 }, /* '253' */
00343 { 214, 214, 214 }, /* '254' */
00344 { 255, 255, 255 }, /* '255' */
00345 { 0, 0, 0 } /* '256' (BYLAYER) */
00346 };
00347
00348
00349 ****
00350 * HUS Colors
00351 ****
00352 const EmbThread husThreads[] = {
00353 {{ 0, 0, 0 }, "Black", "TODO:HUS_CATALOG_NUMBER"}, 
00354 {{ 0, 0, 255 }, "Blue", "TODO:HUS_CATALOG_NUMBER"}, 
00355 {{ 0, 255, 0 }, "Light Green", "TODO:HUS_CATALOG_NUMBER"}, 
00356 {{ 255, 0, 0 }, "Red", "TODO:HUS_CATALOG_NUMBER"}, 
00357 {{ 255, 0, 255 }, "Purple", "TODO:HUS_CATALOG_NUMBER"}, 
00358 {{ 255, 255, 0 }, "Yellow", "TODO:HUS_CATALOG_NUMBER"}, 
00359 {{ 127, 127, 127 }, "Gray", "TODO:HUS_CATALOG_NUMBER"}, 
00360 {{ 51, 154, 255 }, "Light Blue", "TODO:HUS_CATALOG_NUMBER"}, 
00361 {{ 51, 204, 102 }, "Green", "TODO:HUS_CATALOG_NUMBER"}, 
00362 {{ 255, 127, 0 }, "Orange", "TODO:HUS_CATALOG_NUMBER"}, 
00363 {{ 255, 160, 180 }, "Pink", "TODO:HUS_CATALOG_NUMBER"}, 
00364 {{ 153, 75, 0 }, "Brown", "TODO:HUS_CATALOG_NUMBER"}, 
00365 {{ 255, 255, 255 }, "White", "TODO:HUS_CATALOG_NUMBER"}, 
00366 {{ 0, 0, 127 }, "Dark Blue", "TODO:HUS_CATALOG_NUMBER"}, 
00367 {{ 0, 127, 0 }, "Dark Green", "TODO:HUS_CATALOG_NUMBER"}, 
00368 {{ 127, 0, 0 }, "Dark Red", "TODO:HUS_CATALOG_NUMBER"}, 
00369 {{ 255, 127, 127 }, "Light Red", "TODO:HUS_CATALOG_NUMBER"}, 
00370 {{ 127, 0, 127 }, "Dark Purple", "TODO:HUS_CATALOG_NUMBER"}, 
00371 {{ 255, 127, 255 }, "Light Purple", "TODO:HUS_CATALOG_NUMBER"}, 
00372 {{ 200, 200, 0 }, "Dark Yellow", "TODO:HUS_CATALOG_NUMBER"}, 
00373 {{ 255, 255, 153 }, "Light Yellow", "TODO:HUS_CATALOG_NUMBER"}, 
00374 {{ 60, 60, 60 }, "Dark Gray", "TODO:HUS_CATALOG_NUMBER"}, 

```

```
00375 {{ 192, 192, 192 }, "Light Gray", "TODO:HUS_CATALOG_NUMBER"},  
00376 {{ 232, 63, 0 }, "Dark Orange", "TODO:HUS_CATALOG_NUMBER"},  
00377 {{ 255, 165, 65 }, "Light Orange", "TODO:HUS_CATALOG_NUMBER"},  
00378 {{ 255, 102, 122 }, "Dark Pink", "TODO:HUS_CATALOG_NUMBER"},  
00379 {{ 255, 204, 204 }, "Light Pink", "TODO:HUS_CATALOG_NUMBER"},  
00380 {{ 115, 40, 0 }, "Dark Brown", "TODO:HUS_CATALOG_NUMBER"},  
00381 {{ 175, 90, 10 }, "Light Brown", "TODO:HUS_CATALOG_NUMBER"}  
00382 };  
00383  
00384 const EmbThread jefThreads[] = {  
00385 {{0, 0, 0}, "Placeholder", "000"},  
00386 {{0, 0, 0}, "Black", "002"},  
00387 {{255, 255, 255}, "White", "001"},  
00388 {{255, 255, 23}, "Yellow", "204"},  
00389 {{255, 102, 0}, "Orange", "203"},  
00390 {{47, 89, 51}, "Olive Green", "219"},  
00391 {{35, 115, 54}, "Green", "226"},  
00392 {{101, 194, 200}, "Sky", "217"},  
00393 {{171, 90, 150}, "Purple", "208"},  
00394 {{246, 105, 160}, "Pink", "201"},  
00395 {{255, 0, 0}, "Red", "225"},  
00396 {{177, 112, 78}, "Brown", "214"},  
00397 {{11, 47, 132}, "Blue", "207"},  
00398 {{228, 195, 93}, "Gold", "003"},  
00399 {{72, 26, 5}, "Dark Brown", "205"},  
00400 {{172, 156, 199}, "Pale Violet", "209"},  
00401 {{252, 242, 148}, "Pale Yellow", "210"},  
00402 {{249, 153, 183}, "Pale Pink", "211"},  
00403 {{250, 179, 129}, "Peach", "212"},  
00404 {{201, 164, 128}, "Beige", "213"},  
00405 {{151, 5, 51}, "Wine Red", "215"},  
00406 {{160, 184, 204}, "Pale Sky", "216"},  
00407 {{127, 194, 28}, "Yellow Green", "218"},  
00408 {{229, 229, 229}, "Silver Gray", "220"},  
00409 {{136, 155, 155}, "Gray", "221"},  
00410 {{152, 214, 189}, "Pale Aqua", "227"},  
00411 {{178, 225, 227}, "Baby Blue", "228"},  
00412 {{54, 139, 160}, "Powder Blue", "229"},  
00413 {{79, 131, 171}, "Bright Blue", "230"},  
00414 {{56, 106, 145}, "Slate Blue", "231"},  
00415 {{7, 22, 80}, "Navy Blue", "232"},  
00416 {{249, 153, 162}, "Salmon Pink", "233"},  
00417 {{249, 103, 107}, "Coral", "234"},  
00418 {{227, 49, 31}, "Burnt Orange", "235"},  
00419 {{226, 161, 136}, "Cinnamon", "236"},  
00420 {{181, 148, 116}, "Umber", "237"},  
00421 {{228, 207, 153}, "Blond", "238"},  
00422 {{255, 203, 0}, "Sunflower", "239"},  
00423 {{225, 173, 212}, "Orchid Pink", "240"},  
00424 {{195, 0, 126}, "Peony Purple", "241"},  
00425 {{128, 0, 75}, "Burgundy", "242"},  
00426 {{84, 5, 113}, "Royal Purple", "243"},  
00427 {{177, 5, 37}, "Cardinal Red", "244"},  
00428 {{202, 224, 192}, "Opal Green", "245"},  
00429 {{137, 152, 86}, "Moss Green", "246"},  
00430 {{92, 148, 26}, "Meadow Green", "247"},  
00431 {{0, 49, 20}, "Dark Green", "248"},  
00432 {{93, 174, 148}, "Aquamarine", "249"},  
00433 {{76, 191, 143}, "Emerald Green", "250"},  
00434 {{0, 119, 114}, "Peacock Green", "251"},  
00435 {{89, 91, 97}, "Dark Gray", "252"},  
00436 {{255, 255, 242}, "Ivory White", "253"},  
00437 {{177, 88, 24}, "Hazel", "254"},  
00438 {{203, 138, 7}, "Toast", "255"},  
00439 {{152, 108, 128}, "Salmon", "256"},  
00440 {{152, 105, 45}, "Cocoa Brown", "257"},  
00441 {{77, 52, 25}, "Sienna", "258"},  
00442 {{76, 51, 11}, "Sepia", "259"},  
00443 {{51, 32, 10}, "Dark Sepia", "260"},  
00444 {{82, 58, 151}, "Violet Blue", "261"},  
00445 {{13, 33, 126}, "Blue Ink", "262"},  
00446 {{30, 119, 172}, "Sola Blue", "263"},  
00447 {{178, 221, 83}, "Green Dust", "264"},  
00448 {{243, 54, 137}, "Crimson", "265"},  
00449 {{222, 100, 158}, "Floral Pink", "266"},  
00450 {{152, 65, 97}, "Wine", "267"},  
00451 {{76, 86, 18}, "Olive Drab", "268"},  
00452 {{76, 136, 31}, "Meadow", "269"},  
00453 {{228, 222, 121}, "Mustard", "270"},  
00454 {{203, 138, 26}, "Yellow Ocher", "271"},  
00455 {{203, 162, 28}, "Old Gold", "272"},  
00456 {{255, 152, 5}, "Honey Dew", "273"},  
00457 {{252, 178, 87}, "Tangerine", "274"},  
00458 {{0xFF, 0xE5, 0x05}, "Canary Yellow", "275"},  
00459 {{0xFO, 0x33, 0x1F}, "Vermilion", "202"},  
00460 {{0x1A, 0x84, 0x2D}, "Bright Green", "206"},  
00461 {{0x38, 0x6C, 0xAE}, "Ocean Blue", "222"},
```

```

00462     {{0xE3, 0xC4, 0xB4}, "Beige Gray", "223"},  

00463     {{0xE3, 0xAC, 0x81}, "Bamboo", "224"}  

00464 };  

00465  

00466 /*****  

00467 * SHV Colors  

00468 *****/  

00469 const int shvThreadCount = 42;  

00470 const EmbThread shvThreads[] = {  

00471 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00472 {{ 0, 0, 255 }, "Blue", "TODO:CATALOG_NUMBER"},  

00473 {{ 51, 204, 102 }, "Green", "TODO:CATALOG_NUMBER"},  

00474 {{ 255, 0, 0 }, "Red", "TODO:CATALOG_NUMBER"},  

00475 {{ 255, 0, 255 }, "Purple", "TODO:CATALOG_NUMBER"},  

00476 {{ 255, 255, 0 }, "Yellow", "TODO:CATALOG_NUMBER"},  

00477 {{ 127, 127, 127 }, "Grey", "TODO:CATALOG_NUMBER"},  

00478 {{ 51, 154, 255 }, "Light Blue", "TODO:CATALOG_NUMBER"},  

00479 {{ 0, 255, 0 }, "Light Green", "TODO:CATALOG_NUMBER"},  

00480 {{ 255, 127, 0 }, "Orange", "TODO:CATALOG_NUMBER"},  

00481 {{ 255, 160, 180 }, "Pink", "TODO:CATALOG_NUMBER"},  

00482 {{ 153, 75, 0 }, "Brown", "TODO:CATALOG_NUMBER"},  

00483 {{ 255, 255, 255 }, "White", "TODO:CATALOG_NUMBER"},  

00484 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00485 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00486 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00487 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00488 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00489 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00490 {{ 255, 127, 127 }, "Light Red", "TODO:CATALOG_NUMBER"},  

00491 {{ 255, 127, 255 }, "Light Purple", "TODO:CATALOG_NUMBER"},  

00492 {{ 255, 255, 153 }, "Light Yellow", "TODO:CATALOG_NUMBER"},  

00493 {{ 192, 192, 192 }, "Light Grey", "TODO:CATALOG_NUMBER"},  

00494 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00495 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00496 {{ 255, 165, 65 }, "Light Orange", "TODO:CATALOG_NUMBER"},  

00497 {{ 255, 204, 204 }, "Light Pink", "TODO:CATALOG_NUMBER"},  

00498 {{ 175, 90, 10 }, "Light Brown", "TODO:CATALOG_NUMBER"},  

00499 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00500 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00501 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00502 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00503 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00504 {{ 0, 0, 127 }, "Dark Blue", "TODO:CATALOG_NUMBER"},  

00505 {{ 0, 127, 0 }, "Dark Green", "TODO:CATALOG_NUMBER"},  

00506 {{ 127, 0, 0 }, "Dark Red", "TODO:CATALOG_NUMBER"},  

00507 {{ 127, 0, 127 }, "Dark Purple", "TODO:CATALOG_NUMBER"},  

00508 {{ 200, 200, 0 }, "Dark Yellow", "TODO:CATALOG_NUMBER"},  

00509 {{ 60, 60, 60 }, "Dark Gray", "TODO:CATALOG_NUMBER"},  

00510 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00511 {{ 0, 0, 0 }, "Black", "TODO:CATALOG_NUMBER"},  

00512 {{ 232, 63, 0 }, "Dark Orange", "TODO:CATALOG_NUMBER"},  

00513 {{ 255, 102, 122 }, "Dark Pink", "TODO:CATALOG_NUMBER"}  

00514 };  

00515  

00516 const EmbThread pcmThreads[] = {  

00517 {{0x00, 0x00, 0x00}, "PCM Color 1", ""},  

00518 {{0x00, 0x00, 0x80}, "PCM Color 2", ""},  

00519 {{0x00, 0x00, 0xFF}, "PCM Color 3", ""},  

00520 {{0x00, 0x80, 0x80}, "PCM Color 4", ""},  

00521 {{0x00, 0xFF, 0xFF}, "PCM Color 5", ""},  

00522 {{0x80, 0x00, 0x80}, "PCM Color 6", ""},  

00523 {{0xFF, 0x00, 0xFF}, "PCM Color 7", ""},  

00524 {{0x80, 0x00, 0x00}, "PCM Color 8", ""},  

00525 {{0xFF, 0x00, 0x00}, "PCM Color 9", ""},  

00526 {{0x00, 0x80, 0x00}, "PCM Color 10", ""},  

00527 {{0x00, 0xFF, 0x00}, "PCM Color 11", ""},  

00528 {{0x80, 0x80, 0x00}, "PCM Color 12", ""},  

00529 {{0xFF, 0xFF, 0x00}, "PCM Color 13", ""},  

00530 {{0x80, 0x80, 0x80}, "PCM Color 14", ""},  

00531 {{0xC0, 0xC0, 0xC0}, "PCM Color 15", ""},  

00532 {{0xFF, 0xFF, 0xFF}, "PCM Color 16", ""}  

00533 };  

00534  

00535 const int pecThreadCount = 65;  

00536 const EmbThread pecThreads[] = {  

00537 {{ 0, 0, 0 }, "Unknown", "", /* Index 0 */  

00538 {{ 14, 31, 124 }, "Prussian Blue", "", /* Index 1 */  

00539 {{ 10, 85, 163 }, "Blue", "", /* Index 2 */  

00540 /* TODO: Verify RGB value is correct */  

00541 {{ 0, 135, 119 }, "Teal Green", "", /* Index 3 */  

00542 {{ 75, 107, 175 }, "Cornflower Blue", "", /* Index 4 */  

00543 {{ 237, 23, 31 }, "Red", "", /* Index 5 */  

00544 {{ 209, 92, 0 }, "Reddish Brown", "", /* Index 6 */  

00545 {{ 145, 54, 151 }, "Magenta", "", /* Index 7 */  

00546 {{ 228, 154, 203 }, "Light Lilac", "", /* Index 8 */  

00547 {{ 145, 95, 172 }, "Lilac", "", /* Index 9 */  

00548 /* TODO: Verify RGB value is correct */

```

```

00549 {{158, 214, 125}, "Mint Green",      ""}, /* Index 10 */
00550 {{232, 169, 0}, "Deep Gold",        ""}, /* Index 11 */
00551 {{254, 186, 53}, "Orange",          ""}, /* Index 12 */
00552 {{255, 255, 0}, "Yellow",           ""}, /* Index 13 */
00553 {{112, 188, 31}, "Lime Green",      ""}, /* Index 14 */
00554 {{186, 152, 0}, "Brass",            ""}, /* Index 15 */
00555 {{168, 168, 168}, "Silver",          ""}, /* Index 16 */
00556 /* TODO: Verify RGB value is correct */
00557 {{125, 111, 0}, "Russet Brown",     ""}, /* Index 17 */
00558 {{255, 255, 179}, "Cream Brown",    ""}, /* Index 18 */
00559 {{79, 85, 86}, "Pewter",           ""}, /* Index 19 */
00560 {{0, 0, 0}, "Black",                ""}, /* Index 20 */
00561 {{11, 61, 145}, "Ultramarine",     ""}, /* Index 21 */
00562 {{119, 1, 118}, "Royal Purple",    ""}, /* Index 22 */
00563 {{41, 49, 51}, "Dark Gray",        ""}, /* Index 23 */
00564 {{42, 19, 1}, "Dark Brown",        ""}, /* Index 24 */
00565 {{246, 74, 138}, "Deep Rose",       ""}, /* Index 25 */
00566 {{178, 118, 36}, "Light Brown",     ""}, /* Index 26 */
00567 /* TODO: Verify RGB value is correct */
00568 {{252, 187, 197}, "Salmon Pink",    ""}, /* Index 27 */
00569 {{254, 55, 15}, "Vermillion",       ""}, /* Index 28 */
00570 {{240, 240, 240}, "White",          ""}, /* Index 29 */
00571 {{106, 28, 138}, "Violet",          ""}, /* Index 30 */
00572 {{168, 221, 196}, "Seacrest",       ""}, /* Index 31 */
00573 {{37, 132, 187}, "Sky Blue",        ""}, /* Index 32 */
00574 {{254, 179, 67}, "Pumpkin",         ""}, /* Index 33 */
00575 {{255, 243, 107}, "Cream Yellow",   ""}, /* Index 34 */
00576 {{208, 166, 96}, "Khaki",           ""}, /* Index 35 */
00577 {{209, 84, 0}, "Clay Brown",        ""}, /* Index 36 */
00578 {{102, 186, 73}, "Leaf Green",      ""}, /* Index 37 */
00579 {{19, 74, 70}, "Peacock Blue",     ""}, /* Index 38 */
00580 {{135, 135, 135}, "Gray",            ""}, /* Index 39 */
00581 /* TODO: Verify RGB value is correct */
00582 {{216, 204, 198}, "Warm Gray",      ""}, /* Index 40 */
00583 {{67, 86, 7}, "Dark Olive",         ""}, /* Index 41 */
00584 /* TODO: Verify RGB value is correct */
00585 {{253, 217, 222}, "Flesh Pink",     ""}, /* Index 42 */
00586 {{249, 147, 188}, "Pink",            ""}, /* Index 43 */
00587 {{0, 56, 34}, "Deep Green",         ""}, /* Index 44 */
00588 {{178, 175, 212}, "Lavender",        ""}, /* Index 45 */
00589 {{104, 106, 176}, "Wisteria Violet", ""}, /* Index 46 */
00590 {{239, 227, 185}, "Beige",           ""}, /* Index 47 */
00591 {{247, 56, 102}, "Carmine",         ""}, /* Index 48 */
00592 /* TODO: Verify RGB value is correct */
00593 {{181, 75, 100}, "Amber Red",       ""}, /* Index 49 */
00594 {{19, 43, 26}, "Olive Green",       ""}, /* Index 50 */
00595 /* TODO: Verify RGB value is correct */
00596 {{199, 1, 86}, "Dark Fuschia",      ""}, /* Index 51 */
00597 {{254, 158, 50}, "Tangerine",       ""}, /* Index 52 */
00598 {{168, 222, 235}, "Light Blue",      ""}, /* Index 53 */
00599 /* TODO: Verify RGB value is correct */
00600 {{0, 103, 62}, "Emerald Green",     ""}, /* Index 54 */
00601 {{78, 41, 144}, "Purple",           ""}, /* Index 55 */
00602 {{47, 126, 32}, "Moss Green",       ""}, /* Index 56 */
00603 /* TODO: Verify RGB value is correct */
00604 /* TODO: Flesh Pink is Index 42, is this Index incorrect? */
00605 {{255, 204, 204}, "Flesh Pink",     ""}, /* Index 57 */
00606 {{255, 217, 17}, "Harvest Gold",    ""}, /* Index 58 */
00607 {{9, 91, 166}, "Electric Blue",     ""}, /* Index 59 */
00608 {{240, 249, 112}, "Lemon Yellow",   ""}, /* Index 60 */
00609 {{227, 243, 91}, "Fresh Green",     ""}, /* Index 61 */
00610 /* TODO: Verify RGB value is correct */
00611 /* TODO: Orange is Index 12, is this Index incorrect? */
00612 {{255, 153, 0}, "Orange",           ""}, /* Index 62 */
00613 /* TODO: Verify RGB value is correct */
00614 /* TODO: Cream Yellow is Index 34, is this Index incorrect? */
00615 {{255, 240, 141}, "Cream Yellow",   ""}, /* Index 63 */
00616 {{255, 200, 200}, "Applique",       ""}, /* Index 64 */
00617 };
00618
00625 thread_color svg_color_codes[] = {
00626 {"aliceblue", 0xFFf0f8ff, 0},
00627 {"antiquewhite", 0xFFFFfaebd7, 1},
00628 {"aqua", 0xFF00ffff, 2},
00629 {"aquamarine", 0xFF7ffffd4, 3},
00630 {"azure", 0xFFf0ffff, 4},
00631 {"beige", 0xFFFFf5dc, 5},
00632 {"bisque", 0xFFFFfe4c4, 6},
00633 {"black", 0xFF000000, 7},
00634 {"blanchedalmond", 0xFFFFfebcd, 8},
00635 {"blue", 0xFF0000ff, 9},
00636 {"blueviolet", 0xFF8a2be2, 10},
00637 {"brown", 0xFFa52a2a, 11},
00638 {"burlywood", 0xFFddeb81b, 12},
00639 {"cadetblue", 0xFF5f9ea0, 13},
00640 {"chartreuse", 0xFF7fff00, 14},
00641 {"chocolate", 0xFFd22d1e, 15},

```

```
00642 {"coral", 0xFFFF7f0e, 16},
00643 {"cornflowerblue", 0xFFfcfed, 17},
00644 {"cornsilk", 0xFFFFf8dc, 18},
00645 {"crimson", 0xFFdc0e3c, 19},
00646 {"cyan", 0xFF00ffff, 20},
00647 {"darkblue", 0xFF00008b, 21},
00648 {"darkcyan", 0xFF008b8b, 22},
00649 {"darkgoldenrod", 0xFFb81a0b, 23},
00650 {"darkgray", 0xFFa9a9a9, 24},
00651 {"darkgreen", 0xFF001c00, 25},
00652 {"darkgrey", 0xFFa9a9a9, 26},
00653 {"darkkhaki", 0xFFbdb76b, 27},
00654 {"darkmagenta", 0xFF8b008b, 28},
00655 {"darkolivegreen", 0xFF0d6b2f, 29},
00656 {"darkorange", 0xFFFF8c00, 30},
00657 {"darkorchid", 0xFF3f0ecc, 31},
00658 {"darkred", 0xFF8b0000, 32},
00659 {"darksalmon", 0xFFe93c7a, 33},
00660 {"darkseagreen", 0xFF8fb8f8, 34},
00661 {"darkslateblue", 0xFF1e3d8b, 35},
00662 {"darkslategray", 0xFF2f4f4f, 36},
00663 {"darkslategrey", 0xFF2f4f4f, 37},
00664 {"darkturquoise", 0xFF00ced1, 38},
00665 {"darkviolet", 0xFF5e00d3, 39},
00666 {"deeppink", 0xFFff0e5d, 40},
00667 {"deepskyblue", 0xFF00bfff, 41},
00668 {"dimgray", 0xFF2d2d2d, 42},
00669 {"imgrey", 0xFF2d2d2d, 43},
00670 {"dodgerblue", 0xFF1e5aff, 44},
00671 {"firebrick", 0xFFb20a0a, 45},
00672 {"floralwhite", 0xFFFFfa0, 46},
00673 {"forestgreen", 0xFF0a8b0a, 47},
00674 {"fuchsia", 0xFFFF00ff, 48},
00675 {"gainsboro", 0xFFcdcdc, 49},
00676 {"ghostwhite", 0xFFFFff8ff, 50},
00677 {"gold", 0xFFFFfd700, 51},
00678 {"goldenrod", 0xFFdaa50e, 52},
00679 {"gray", 0xFF0e0e0e, 53},
00680 {"grey", 0xFF0e0e0e, 54},
00681 {"green", 0xFF000e00, 55},
00682 {"greenyellow", 0xFFadff2f, 56},
00683 {"honeydew", 0xFFFF0fff0, 57},
00684 {"hotpink", 0xFFFF2db4, 58},
00685 {"indianred", 0xFFcd5c5c, 59},
00686 {"indigo", 0xFF4b000a, 60},
00687 {"ivory", 0xFFFFFFFF, 61},
00688 {"khaki", 0xFF0e68c, 62},
00689 {"lavender", 0xFFe6e6fa, 63},
00690 {"lavenderblush", 0xFFFFf0f5, 64},
00691 {"lawngreen", 0xFF7cfcc0, 65},
00692 {"lemonchiffon", 0xFFFFfacd, 66},
00693 {"lightblue", 0xFFadd8e6, 67},
00694 {"lightcoral", 0xFF00e0e, 68},
00695 {"lightcyan", 0xFFe0ffff, 69},
00696 {"lightgoldenrodyellow", 0xFFfafad2, 70},
00697 {"lightgray", 0xFFd3d3d3, 71},
00698 {"lightgreen", 0xFF5aae5a, 72},
00699 {"lightgrey", 0xFFd3d3d3, 73},
00700 {"lightpink", 0xFFfb6c1, 74},
00701 {"lightsalmon", 0xFFfa07a, 75},
00702 {"lightseagreen", 0xFF0eb2aa, 76},
00703 {"lightskyblue", 0xFF1bcef8, 77},
00704 {"lightslategray", 0xFF4d3a3f, 78},
00705 {"lightslategrey", 0xFF4d3a3f, 79},
00706 {"lightsteelblue", 0xFFb0c4de, 80},
00707 {"lightyellow", 0xFFFFffe0, 81},
00708 {"lime", 0xFF00ff00, 82},
00709 {"limegreen", 0xFF0ecd0e, 83},
00710 {"linen", 0xFFfaf0e6, 84},
00711 {"magenta", 0xFFFF00ff, 85},
00712 {"maroon", 0xFF0e0000, 86},
00713 {"mediumaquamarine", 0xFF2acdaa, 87},
00714 {"mediumblue", 0xFF0000cd, 88},
00715 {"mediumorchid", 0xFFba0dd3, 89},
00716 {"mediumpurple", 0xFF5d2edb, 90},
00717 {"mediumseagreen", 0xFF3cb32f, 91},
00718 {"mediumslateblue", 0xFF7b2cee, 92},
00719 {"mediumspringgreen", 0xFF00fa9a, 93},
00720 {"mediumturquoise", 0xFF1ed1cc, 94},
00721 {"mediumvioletred", 0xFFc70f0d, 95},
00722 {"midnightblue", 0xFF0d0d2e, 96},
00723 {"mintcream", 0xFFf5ffa, 97},
00724 {"mistyrose", 0xFFffe4e1, 98},
00725 {"moccasin", 0xFFffe4b5, 99},
00726 {"navajowhite", 0xFFFFdead, 100},
00727 {"navy", 0xFF00000e, 101},
00728 {"oldlace", 0xFFfdf5e6, 102},
```

```

00729 {"olive", 0xFF0e0e00, 103},
00730 {"olivedrab", 0xFF6b8e0b, 104},
00731 {"orange", 0xFFFFfa500, 105},
00732 {"orangered", 0xFFFFf2d00, 106},
00733 {"orchid", 0xFFda2ed6, 107},
00734 {"palegoldenrod", 0xFFee8aa, 109},
00735 {"palegreen", 0xFF3efb3e, 110},
00736 {"paleturquoise", 0xFFafeeee, 111},
00737 {"palevioletred", 0xFFdb2e5d, 112},
00738 {"papayawhip", 0xFFffed5, 113},
00739 {"peachpuff", 0xFFFFdab9, 114},
00740 {"peru", 0xFFcd0d3f, 115},
00741 {"pink", 0xFFFFfc0cb, 116},
00742 {"plum", 0xFFdda0dd, 117},
00743 {"powderblue", 0xFFb0e0e6, 118},
00744 {"purple", 0xFF0e000e, 119},
00745 {"red", 0xFFFFf0000, 120},
00746 {"rosybrown", 0xFFbc8f8f, 121},
00747 {"royalblue", 0xFF1d2de1, 122},
00748 {"saddlebrown", 0xFF8b2d0d, 123},
00749 {"salmon", 0xFFfa0ele, 124},
00750 {"sandybrown", 0xFFFFd4a43c, 125},
00751 {"seagreen", 0xFF2e8b1b, 126},
00752 {"seashell", 0xFFFFf5ee, 127},
00753 {"sienna", 0xFFa0a2d, 128},
00754 {"silver", 0xFFC0c0c0, 129},
00755 {"skyblue", 0xFF1bceeb, 130},
00756 {"slateblue", 0xFF6a5acd, 131},
00757 {"slategray", 0xFF2e0e5a, 132},
00758 {"slategrey", 0xFF2e0e5a, 133},
00759 {"snow", 0xFFFFfafa, 134},
00760 {"springgreen", 0xFF00ff7f, 135},
00761 {"steelblue", 0xFF2e0ab4, 136},
00762 {"tan", 0xFFd2b48c, 137},
00763 {"teal", 0xFF000e0e, 138},
00764 {"thistle", 0xFFd8bfd8, 139},
00765 {"tomato", 0xFFFFf3f2f, 140},
00766 {"turquoise", 0xFF1ce0d0, 141},
00767 {"violet", 0xFFFFee0aee, 142},
00768 {"wheat", 0xFFf5deb3, 143},
00769 {"white", 0xFFFFffff, 144},
00770 {"whitesmoke", 0xFFFF5f5f5, 145},
00771 {"yellow", 0xFFFFffff00, 146},
00772 {"yellowgreen", 0xFF9acd0e, 147},
00773 {"END", 0, -1}
00774 };
00775
00776 thread_color Arc_Polyester_codes[] = {
00777 {"END", 0, -1}
00778 };
00779
00780 thread_color Arc_Rayon_codes[] = {
00781 {"END", 0, -1}
00782 };
00783
00784 thread_color CoatsAndClark_Rayon_codes[] = {
00785 {"END", 0, -1}
00786 };
00787
00788 thread_color Exquisite_Polyester_codes[] = {
00789 {"END", 0, -1}
00790 };
00791
00792 thread_color Fufu_Polyester_codes[] = {
00793 {"END", 0, -1}
00794 };
00795
00796 thread_color Fufu_Rayon_codes[] = {
00797 {"END", 0, -1}
00798 };
00799
00800 thread_color Hemingworth_Polyester_codes[] = {
00801 {"Pure White", 0xFFFFFFFF, 1001},
00802 {"Lemon Ice", 0xFFDDE00F, 1271},
00803 {"Neon Green", 0xFFC9DD03, 1272},
00804 {"Brilliant Lime", 0xFF60DD49, 1273},
00805 {"Mango", 0xFFFFCC1E, 1274},
00806 {"Neon Yellow", 0xFFFFED38, 1275},
00807 {"Tropical Orange", 0xFFFFA952, 1276},
00808 {"Neon Orange", 0xFFFF9338, 1277},
00809 {"Rebel Peach", 0xFFFF585F, 1278},
00810 {"Shy Flamingo", 0xFFF28CA3, 1279},
00811 {"Neon Pink", 0xFFFFE8A9E, 1280},
00812 {"Neon Peach", 0xFFFFC074F, 1281},
00813 {"Dove Gray", 0xFFFCFC3C3, 1067},
00814 {"Silver Lining", 0xFFC9CAC8, 1068},
00815 {"Storm Cloud", 0xFFB2B4B3, 1069},

```

```
00816 {"Platinum", 0xFFC6C6BC, 1070},  
00817 {"Graphite", 0xFF616365, 1244},  
00818 {"Light Charcoal", 0xFF4D4F53, 1245},  
00819 {"Chrome", 0xFF8E908F, 1072},  
00820 {"Antique Silver", 0xFF747678, 1077},  
00821 {"Pewter Gray", 0xFF6C6F70, 1073},  
00822 {"Black Stallion", 0xFF191D1F, 1079},  
00823 {"Charcoal", 0xFF1B242A, 1087},  
00824 {"Classic Black", 0xFF000000, 1000},  
00825 {"Marshmallow", 0xFFFFD5D2CA, 1118},  
00826 {"Ice Blue", 0xFFDAE3EA, 1119},  
00827 {"Nautical Blue", 0xFFA6BCC6, 1076},  
00828 {"Sea Storm", 0xFF818A8F, 1074},  
00829 {"Bronzed Steel", 0xFF595A5C, 1078},  
00830 {"Silvery Gray", 0xFFB9C9D0, 1239},  
00831 {"Granite", 0xFF7D9AAA, 1240},  
00832 {"Shadow", 0xFFFF5E6A71, 1085},  
00833 {"Dark Slate Blue", 0xFF003C69, 1241},  
00834 {"Deep Slate Blue", 0xFF003946, 1242},  
00835 {"Pacific Waters", 0xFF004250, 1081},  
00836 {"Dark Slate", 0xFF37424A, 1086},  
00837 {"Smoky Blue", 0xFF005B82, 1192},  
00838 {"Light Slate Blue", 0xFF5E9CAE, 1193},  
00839 {"Hyacinth", 0xFF6AADE4, 1188},  
00840 {"Bluebird", 0xFF4B92DB, 1187},  
00841 {"Misty Blue", 0xFF8FCAE7, 1186},  
00842 {"Cornflower Blue", 0xFFC2DEEA, 1185},  
00843 {"Pale Blue", 0xFFA0CFEB, 1256},  
00844 {"Country Blue", 0xFF0098DB, 1255},  
00845 {"Azure", 0xFF3D7EDB, 1202},  
00846 {"Royal Blue", 0xFF0039A6, 1203},  
00847 {"Brilliant Blue", 0xFF00338D, 1204},  
00848 {"Deep Blue", 0xFF0B2265, 1205},  
00849 {"Winter Blue", 0xFF98C6EA, 1189},  
00850 {"Winter Sky", 0xFFAACAE6, 1190},  
00851 {"Sky Blue", 0xFF8EBAE5, 1191},  
00852 {"China Blue", 0xFF0073CF, 1198},  
00853 {"Dark Blueberry", 0xFF004165, 1201},  
00854 {"Salem Blue", 0xFF004153, 1200},  
00855 {"Navy", 0xFF002244, 1199},  
00856 {"Sailor Blue", 0xFF002C5F, 1265},  
00857 {"Dark Blue ", 0xFF002857, 1264},  
00858 {"Berry Blue", 0xFF003591, 1263},  
00859 {"True Blue", 0xFF002C77, 1261},  
00860 {"Periwinkle", 0xFF6F9AD3, 1262},  
00861 {"Iceberg Blue", 0xFF65CFE9, 1197},  
00862 {"Medium Aquamarine", 0xFF0075B0, 1195},  
00863 {"Dark Aquamarine", 0xFF0066A1, 1196},  
00864 {"Peacock Blue", 0xFF006983, 1194},  
00865 {"Dark Turquoise", 0xFF003D4C, 1258},  
00866 {"Turquoise", 0xFF0098C3, 1259},  
00867 {"Caribbean Blue", 0xFF00B0CA, 1260},  
00868 {"Summer Sky", 0xFF6FD4E4, 1257},  
00869 {"Crystal Lake", 0xFFBBE7E6, 1178},  
00870 {"Icicle Blue", 0xFFC1E2E5, 1172},  
00871 {"Frosty Blue", 0xFF8FDFE2, 1173},  
00872 {"Blue Lagoon", 0xFF00AFD8, 1174},  
00873 {"Blue Satin", 0xFF006778, 1181},  
00874 {"Teal Blue", 0xFF007C92, 1180},  
00875 {"Light Teal Blue", 0xFF009AA6, 1176},  
00876 {"Wintergreen", 0xFF7CA295, 1175},  
00877 {"Mint Green", 0xFF63CECA, 1177},  
00878 {"Navajo Turquoise", 0xFF00877C, 1179},  
00879 {"Peacock Green", 0xFF007B69, 1182},  
00880 {"Forest Glen", 0xFF024E43, 1183},  
00881 {"Deep Teal", 0xFF004953, 1184},  
00882 {"Deep Sea", 0xFF156570, 1082},  
00883 {"Dragonfly", 0xFF00505C, 1113},  
00884 {"Blue Steel", 0xFF44697D, 1084},  
00885 {"Dark Sage", 0xFF496C60, 1114},  
00886 {"Silver Green", 0xFF949D9E, 1115},  
00887 {"Antique Gray", 0xFF91BA43, 1243},  
00888 {"Ocean Spray", 0xFFB9CCC3, 1100},  
00889 {"Sea Foam", 0xFFA6E6BC, 1088},  
00890 {"Cucumber Melon", 0xFF00B588, 1094},  
00891 {"Light Jade", 0xFF00985F, 1106},  
00892 {"Jade", 0xFF009B74, 1107},  
00893 {"Dark Jade", 0xFF007D57, 1105},  
00894 {"Caribbean", 0xFF006A4D, 1104},  
00895 {"Dark Teal", 0xFF00685B, 1254},  
00896 {"Minty Teal", 0xFF0D776E, 1253},  
00897 {"Lemony Lime", 0xFFC3E76F, 1099},  
00898 {"Kiwi Lime", 0xFFCCDC00, 1247},  
00899 {"Electric Green", 0xFF69BE28, 1097},  
00900 {"Green Apple", 0xFF92D400, 1091},  
00901 {"Key Lime", 0xFF7AB800, 1092},  
00902 {"Kelly Green", 0xFF3F9C35, 1093},
```

```
00903 {"Meadow", 0xFF00AF3F, 1248},  
00904 {"Grassy Green", 0xFF007934, 1095},  
00905 {"Dark Kelly Green", 0xFF008542, 1108},  
00906 {"Christmas Green", 0xFF00693C, 1109},  
00907 {"Winter Pine ", 0xFF1C453B, 1250},  
00908 {"Holly Leaf", 0xFF175E54, 1249},  
00909 {"Pistachio Nut", 0xFFC8E59A, 1096},  
00910 {"Dusty Green", 0xFF69923A, 1251},  
00911 {"Bush Ivy ", 0xFF557630, 1252},  
00912 {"Leafy Green", 0xFF739600, 1089},  
00913 {"Kentucky Grass", 0xFF53682B, 1090},  
00914 {"Ivy", 0xFF035642, 1103},  
00915 {"Evergreen", 0xFF284E36, 1110},  
00916 {"Mountain Meadow", 0xFF004438, 1111},  
00917 {"Forest Green", 0xFF004D46, 1112},  
00918 {"Oregano", 0xFF57584F, 1121},  
00919 {"Jungle Green", 0xFF404A29, 1123},  
00920 {"Thyme", 0xFF83847A, 1120},  
00921 {"Light Avocado", 0xFF827C34, 1058},  
00922 {"Split Pea", 0xFFB19B00, 1061},  
00923 {"Spring Leaf", 0xFFB5A300, 1059},  
00924 {"Almond Cream", 0xFFF8E498, 1060},  
00925 {"Eggshell", 0xFFD7D3C7, 1229},  
00926 {"Corn silk Green", 0xFFD5C833, 1098},  
00927 {"Avocado", 0xFF6A7029, 1101},  
00928 {"Seaweed", 0xFF898F4B, 1102},  
00929 {"Olive Green", 0xFF65551C, 1246},  
00930 {"Coconut Shell", 0xFF4B452C, 1117},  
00931 {"Parsley", 0xFF4B471A, 1116},  
00932 {"Dried Sage", 0xFF718674, 1083},  
00933 {"Mocha", 0xFF4F4C25, 1125},  
00934 {"Warm Earth", 0xFF5D4F4B, 1131},  
00935 {"Dark Chocolate", 0xFF452325, 1126},  
00936 {"Deep Walnut", 0xFF4E2E2D, 1128},  
00937 {"Teddybear Brown", 0xFF6E3219, 1130},  
00938 {"Light Chestnut", 0xFF60351D, 1134},  
00939 {"Pecan Pie", 0xFF6C4D23, 1140},  
00940 {"Dark Alder", 0xFF766A65, 1237},  
00941 {"Army Green", 0xFF5B491F, 1137},  
00942 {"Pharaoh Gold", 0xFF6E5A2A, 1135},  
00943 {"Autumn Haystack", 0xFFAB8422, 1136},  
00944 {"Sahara", 0xFF856822, 1122},  
00945 {"Weathered Wood", 0xFF675C53, 1236},  
00946 {"Soft Beige", 0xFF9A996E, 1124},  
00947 {"Mercury", 0xFFC2B2B5, 1149},  
00948 {"Old Lace", 0xFFA5ACAF, 1146},  
00949 {"Caramel Cappuccino", 0xFFAE7D5B, 1145},  
00950 {"Brown Sugar", 0xFFA76F3E, 1133},  
00951 {"Light Cinnamon", 0xFFA25022, 1064},  
00952 {"Cinnamon", 0xFF86431E, 1238},  
00953 {"Apple Cider", 0xFFB2541A, 1163},  
00954 {"Indian Paintbrush", 0xFF9A3B26, 1144},  
00955 {"Rust", 0xFF833820, 1164},  
00956 {"Toasted Almond", 0xFF825C26, 1142},  
00957 {"Pale Caramel", 0xFFC59217, 1063},  
00958 {"Honey Butter", 0xFFDDB99A, 1062},  
00959 {"Sandy Shore", 0xFFD2C295, 1057},  
00960 {"Ecru", 0xFFC2C2A0, 1056},  
00961 {"Malt", 0xFFB3B38C, 1055},  
00962 {"Antique Lace", 0xFFC7B37F, 1054},  
00963 {"Champagne", 0xFFBD9271, 1171},  
00964 {"Butter Taffy", 0xFFB3995D, 1138},  
00965 {"Cream Soda", 0xFFCEA98C, 1235},  
00966 {"Conch Shell", 0xFFE39B6C, 1139},  
00967 {"New Penny", 0xFFBA6F2E, 1132},  
00968 {"Pumpkin Spice", 0xFFBB650E, 1141},  
00969 {"Soft Sunlight", 0xFFEBE8B1, 1042},  
00970 {"Lemon Drop", 0xFFEEEC83, 1043},  
00971 {"Daffodil", 0xFFFF3EC7A, 1045},  
00972 {"Lemon Citrus", 0xFFFF5EC5A, 1225},  
00973 {"Sunshine Yellow", 0xFFFFAE700, 1226},  
00974 {"Canary Yellow", 0xFFFF2EE72, 1044},  
00975 {"Sunflower", 0xFFFFCD900, 1227},  
00976 {"Sun", 0xFFFFADC41, 1046},  
00977 {"Dandelion", 0xFFFFED100, 1047},  
00978 {"Buttercup", 0xFFFF3CF45, 1048},  
00979 {"Ginger Root", 0xFFFFFB047, 1050},  
00980 {"Goldenrod", 0xFFEAAB00, 1051},  
00981 {"Cornsilk", 0xFFDCD6B2, 1037},  
00982 {"Macadamia", 0xFFC6BC89, 1038},  
00983 {"Yellow Plumeria", 0xFFFF8E498, 1039}, /* TODO: duplicate case value */  
00984 {"Maize", 0xFFFF8DE6E, 1040},  
00985 {"Dried Banana", 0xFFFFADA63, 1049},  
00986 {"Butternut", 0xFFFFFCB4F, 1053},  
00987 {"Orange Meringue", 0xFFFFA100, 1232},  
00988 {"September Sunset", 0xFFFFFB612, 1231},  
00989 {"Orange Cream", 0xFFFFFB652, 1230},
```

```
00990 {"Cantaloupe", 0xFFFFFBC3D, 1041},  
00991 {"Old Gold", 0xFFCE8E00, 1052},  
00992 {"Auburn", 0xFF9D5324, 1143},  
00993 {"Citrus Burst", 0xFFE98300, 1024},  
00994 {"Orange Slice", 0xFFFFF7000, 1025},  
00995 {"Fiery Sunset", 0xFFE37222, 1027},  
00996 {"Hunter Orange", 0xFFFB4F14, 1028},  
00997 {"Fall Harvest", 0xFFDD4814, 1029},  
00998 {"Candy Apple", 0xFFCD202C, 1030},  
00999 {"Christmas Red", 0xFFC30014, 1270},  
01000 {"Pomegranate", 0xFFA70232, 1032},  
01001 {"Rummy Raisin", 0xFF882332, 1031},  
01002 {"Cardinal Red", 0xFFA51100, 1002},  
01003 {"Rusty Red", 0xFF9E3039, 1234},  
01004 {"Redwood", 0xFF783014, 1233},  
01005 {"Carrot", 0xFFD55C19, 1065},  
01006 {"Paprika", 0xFFAA272F, 1066},  
01007 {"Cherrywood", 0xFF5F3327, 1129},  
01008 {"Burnt Sienna", 0xFF5D3526, 1127},  
01009 {"Merlot", 0xFF592C35, 1160},  
01010 {"Loganberry", 0xFF6A1A41, 1159},  
01011 {"Cranberry", 0xFFE2714, 1158},  
01012 {"Mulberry", 0xFF662046, 1157},  
01013 {"Magenta", 0xFF85003C, 1156},  
01014 {"Raspberry", 0xFF641F14, 1155},  
01015 {"Salmon", 0xFFFFF818D, 1166},  
01016 {"Georgia Peach", 0xFFFFFA48A, 1015},  
01017 {"Rose Sunset", 0xFFFFFB0B7, 1011},  
01018 {"Bubblegum Pink", 0xFFFF3789B, 1012},  
01019 {"Carnation", 0xFFDB4D69, 1014},  
01020 {"Very Berry", 0xFF91004B, 1013},  
01021 {"Raspberry Red", 0xFF82240C, 1224},  
01022 {"Dark Salmon", 0xFFFF54359, 1018},  
01023 {"Apricot Dream", 0xFFFF4587A, 1017},  
01024 {"Coral Reef", 0xFFFF8B7C, 1016},  
01025 {"Frosted Peach", 0xFFFFFC19C, 1022},  
01026 {"Tangerine", 0xFFFF8F70, 1020},  
01027 {"Dark Mango", 0xFFFF6D42, 1026},  
01028 {"Marigold", 0xFFFFA02F, 1023},  
01029 {"Spun Silk", 0xFFECC182, 1168},  
01030 {"Whipped Papaya", 0xFFFBCE92, 1021},  
01031 {"Baby Peach", 0xFFFFDC480, 1228},  
01032 {"Pink Pearl", 0xFFFFFC2A2, 1167},  
01033 {"Peaches 'n Cream", 0xFFEFC5CE, 1169},  
01034 {"Peach Pastel", 0xFFFFFBEB9C, 1170},  
01035 {"Old Penny", 0xFF774A39, 1162},  
01036 {"Dusty Rose", 0xFFB26F7E, 1151},  
01037 {"Winter Rose", 0xFFD490A8, 1165},  
01038 {"Valentine Pink", 0xFF6A3BB, 1161},  
01039 {"Petal Peach", 0xFFFFFB7AE, 1019},  
01040 {"Soft Petal", 0xFFEF6DB, 1150},  
01041 {"Fuchsia", 0xFF920075, 1036},  
01042 {"Pink Kiss", 0xFFF375C6, 1004},  
01043 {"Baby Pink", 0xFFF3BBC, 1003},  
01044 {"Whisper Pink", 0xFFF1DBDF, 1005},  
01045 {"Gentle Blush", 0xFFF3C9D3, 1006},  
01046 {"English Rose", 0xFFF4B2C1, 1007},  
01047 {"Sweet Pea", 0xFFF39EBC, 1008},  
01048 {"Rosy Blush", 0xFFFF77AB4, 1009},  
01049 {"Passion Pink", 0xFFD71F85, 1010},  
01050 {"Mulled Wine", 0xFF772059, 1035},  
01051 {"Primrose", 0xFFC50084, 1034},  
01052 {"Azalea", 0xFFA1006B, 1033},  
01053 {"Snowflake", 0xFFD1D4D3, 1148},  
01054 {"Moonlight", 0xFFCAD1E7, 1147},  
01055 {"Tulip", 0xFF9DABE2, 1207},  
01056 {"Purple Iris", 0xFF8884D5, 1206},  
01057 {"Grape", 0xFF1A2155, 1209},  
01058 {"Moon Shadow", 0xFF6459C4, 1211},  
01059 {"Electric Purple", 0xFF212492, 1208},  
01060 {"Indigo", 0xFF411244, 1210},  
01061 {"Royal Purple", 0xFF3B0083, 1223},  
01062 {"Eggplant", 0xFF151C54, 1267},  
01063 {"Dark Purple", 0xFF490E6F, 1269},  
01064 {"Pure Purple", 0xFF57068C, 1268},  
01065 {"Pale Orchid", 0xFFDC7DF, 1217},  
01066 {"Dusty Mauve", 0xFFC2ACBE, 1219},  
01067 {"Orchid", 0xFFDC9DDD, 1218},  
01068 {"Heather", 0xFFB382C7, 1213},  
01069 {"Lavender", 0xFF9C5FB5, 1214},  
01070 {"Soft Grape", 0xFF4B08A1, 1266},  
01071 {"Freesia", 0xFFC1AFE5, 1221},  
01072 {"Lilac", 0xFFC5B9E3, 1222},  
01073 {"Peony", 0xFF6E2C6B, 1215},  
01074 {"Dark Fuschia", 0xFF7D0063, 1216},  
01075 {"Grape Jelly", 0xFF752864, 1220},  
01076 {"Deep Orchid", 0xFF55517B, 1080},
```

```
01077 {"Misty Blue Gray", 0xFF5C7F92, 1075},  
01078 {"Iron Ore", 0xFFAFADC3, 1071},  
01079 {"Light Mauve", 0xFFD8AAB3, 1152},  
01080 {"Dark Mauve", 0xFF89687C, 1153},  
01081 {"Wild Plum", 0xFF644459, 1154},  
01082 {"Huckleberry", 0xFF4B306A, 1212},  
01083 {"END", 0, -1}  
01084 };  
01085  
01086 thread_color Isacord_Polyester_codes[] = {  
01087 {"?", 0xFFFFFFFF, 10},  
01088 {"?", 0xFFFFFFFF, 15}, /* TODO: duplicate case value */  
01089 {"?", 0xFFFFFFFF, 17}, /* TODO: duplicate case value */  
01090 {"?", 0x00000000, 20},  
01091 {"?", 0xFFFFFDDED, 101},  
01092 {"?", 0xFF6D757B, 108},  
01093 {"?", 0xFF515B61, 111},  
01094 {"?", 0xFF5D5D5D, 112},  
01095 {"?", 0xFFCFCCFCF, 124},  
01096 {"?", 0xFFA1A9B4, 131},  
01097 {"?", 0xFF192024, 132},  
01098 {"?", 0xFF9EA5AA, 142},  
01099 {"?", 0xFFCFD1D5, 145},  
01100 {"?", 0xFFC6BDB4, 150},  
01101 {"?", 0xFFD5C4B3, 151},  
01102 {"?", 0xFF7C8283, 152},  
01103 {"?", 0xFFFFEF5F0, 180},  
01104 {"?", 0xFFE9D7D9, 182},  
01105 {"?", 0xFFEBE3DD, 184},  
01106 {"?", 0xFFE0DA5F, 221},  
01107 {"?", 0xFFFFFBBA28, 232},  
01108 {"?", 0xFFFFAF6CC, 250},  
01109 {"?", 0xFFFF9F8E8, 270},  
01110 {"?", 0xFFFFDF76C, 310},  
01111 {"?", 0xFFFF5D300, 311},  
01112 {"?", 0xFF797E24, 345},  
01113 {"?", 0xFFB0AA76, 352},  
01114 {"?", 0xFF898F2B, 442},  
01115 {"?", 0xFF98996D, 453},  
01116 {"?", 0xFF6B7E6F, 463},  
01117 {"?", 0xFF3E4F34, 465},  
01118 {"?", 0FFEDEF05, 501},  
01119 {"?", 0xFFFF5D300, 506}, /* TODO: duplicate case value */  
01120 {"?", 0xFFFFDE896, 520},  
01121 {"?", 0xFFD7CE8A, 532},  
01122 {"?", 0xFFB18B00, 542},  
01123 {"?", 0xFFB28F11, 546},  
01124 {"?", 0xFFB69F56, 552},  
01125 {"?", 0xFFFF8D73B, 600},  
01126 {"?", 0xFFFF8D73E, 605}, /* TODO: duplicate case value */  
01127 {"?", 0xFFFF7DC00, 608},  
01128 {"?", 0xFFFFEF09A, 630},  
01129 {"?", 0xFFFFDE896, 640}, /* TODO: duplicate case value */  
01130 {"?", 0xFFFF5D2A6, 651},  
01131 {"?", 0xFFFFF9EA, 660},  
01132 {"?", 0xFFFFAF6E7, 670},  
01133 {"?", 0xFFEBEBEA8, 672},  
01134 {"?", 0xFFFF7C35F, 700},  
01135 {"?", 0xFFFF5CA00, 702},  
01136 {"?", 0xFFFFDFA200, 704},  
01137 {"?", 0xFFFFCF538, 706},  
01138 {"?", 0xFFFFADC59, 713},  
01139 {"?", 0xFF8C7E6A, 722},  
01140 {"?", 0xFF594900, 747},  
01141 {"?", 0xFFD6BF94, 761},  
01142 {"?", 0xFF656452, 776},  
01143 {"?", 0xFFF1AF00, 800},  
01144 {"?", 0xFFFF5BA5D, 811},  
01145 {"?", 0FFE1A23E, 821},  
01146 {"?", 0xFFCCAB3F, 822},  
01147 {"?", 0xFFFFDFA200, 824}, /* TODO: duplicate case value */  
01148 {"?", 0xFFD0A44F, 832},  
01149 {"?", 0xFFCD944A, 842},  
01150 {"?", 0xFFE3BC61, 851},  
01151 {"?", 0xFF947C4A, 853},  
01152 {"?", 0xFFCBFBFA2, 861},  
01153 {"?", 0xFFA5866A, 862},  
01154 {"?", 0xFFEBE7DD, 870},  
01155 {"?", 0xFF9FA086, 873},  
01156 {"?", 0xFF9A897B, 874},  
01157 {"?", 0xFFFF3B259, 904},  
01158 {"?", 0FFCA832C, 922},  
01159 {"?", 0FFC07314, 931},  
01160 {"?", 0FFAC6613, 932},  
01161 {"?", 0FF744808, 933},  
01162 {"?", 0FFBD9565, 934},  
01163 {"?", 0FFC98300, 940},
```

```
01164 {"?", 0xFFAF7D3E, 941},  
01165 {"?", 0xFF483928, 945},  
01166 {"?", 0xFFFFEFEED, 970},  
01167 {"?", 0xFF6A4129, 1055},  
01168 {"?", 0xFFFDE2C1, 1060},  
01169 {"?", 0xFFA68A68, 1061},  
01170 {"?", 0xFFED9206, 1102},  
01171 {"?", 0xFFEE9C00, 1106},  
01172 {"?", 0xFFE8751, 1114},  
01173 {"?", 0xFFA35214, 1115},  
01174 {"?", 0xFFF8C000, 1120},  
01175 {"?", 0xFFB7976B, 1123},  
01176 {"?", 0xFF9D5A21, 1134},  
01177 {"?", 0xFFF3D8A8, 1140},  
01178 {"?", 0xFFFFACFAE, 1141},  
01179 {"?", 0xFF7A4427, 1154},  
01180 {"?", 0xFFDFC8AB, 1172},  
01181 {"?", 0xFFE89763, 1211},  
01182 {"?", 0xFFF1A236, 1220},  
01183 {"?", 0xFFE5571D, 1300},  
01184 {"?", 0xFFD9674C, 1301},  
01185 {"?", 0xFFE4501E, 1304},  
01186 {"?", 0xFFEA7134, 1305},  
01187 {"?", 0xFFE12A23, 1306},  
01188 {"?", 0xFFC14817, 1311},  
01189 {"?", 0xFFC45331, 1312},  
01190 {"?", 0xFFD5815E, 1332},  
01191 {"?", 0xFFBB3D2E, 1334},  
01192 {"?", 0xFFBE2D1A, 1335},  
01193 {"?", 0xFF5F1B23, 1342},  
01194 {"?", 0xFF7A3441, 1346},  
01195 {"?", 0xFFFFBF95, 1351},  
01196 {"?", 0xFFF4A773, 1352},  
01197 {"?", 0xFF693920, 1355},  
01198 {"?", 0xFF9C598, 1362},  
01199 {"?", 0xFF432731, 1366},  
01200 {"?", 0xFF464537, 1375},  
01201 {"?", 0xFFF4A782, 1430},  
01202 {"?", 0xFFE22D2A, 1501},  
01203 {"?", 0xFFA93121, 1514},  
01204 {"?", 0xFFEC7168, 1521},  
01205 {"?", 0xFFF6B08E, 1532},  
01206 {"?", 0xFFF9C5B9, 1551},  
01207 {"?", 0xFF806A61, 1565},  
01208 {"?", 0FFE36C63, 1600},  
01209 {"?", 0FFE44733, 1701},  
01210 {"?", 0FFDFO032, 1703},  
01211 {"?", 0FFE0003D, 1704},  
01212 {"?", 0FFCF0040, 1725},  
01213 {"?", 0FFF1CDCE, 1755},  
01214 {"?", 0FFE9C9BD, 1760},  
01215 {"?", 0FFE8C0B8, 1761},  
01216 {"?", 0FFE00046, 1800},  
01217 {"?", 0FFD6445D, 1805},  
01218 {"?", 0FFF49E95, 1840},  
01219 {"?", 0FFFCDA5, 1860},  
01220 {"?", 0FF636254, 1874},  
01221 {"?", 0FF394535, 1876},  
01222 {"?", 0FFE10057, 1900},  
01223 {"?", 0FFBD0041, 1902},  
01224 {"?", 0FFC00343, 1903},  
01225 {"?", 0FFA9023A, 1904},  
01226 {"?", 0FFBE004F, 1906},  
01227 {"?", 0FF910230, 1911},  
01228 {"?", 0FF86023E, 1912},  
01229 {"?", 0FF9A0C3B, 1913},  
01230 {"?", 0FFA33050, 1921},  
01231 {"?", 0FFF28DA6, 1940},  
01232 {"?", 0FFCE427A, 1950},  
01233 {"?", 0FF959595, 1972},  
01234 {"?", 0FFA33145, 2011},  
01235 {"?", 0FF9F454C, 2022},  
01236 {"?", 0FFC7979B, 2051},  
01237 {"?", 0FF9F003F, 2101},  
01238 {"?", 0FF78093F, 2113},  
01239 {"?", 0FF6D0627, 2115},  
01240 {"?", 0FF432732, 2123},  
01241 {"?", 0FFE6778B, 2152},  
01242 {"?", 0FFDF8390, 2153},  
01243 {"?", 0FFF9BFC0, 2155},  
01244 {"?", 0FFFB1D6, 2160},  
01245 {"?", 0FFD8D5D0, 2166},  
01246 {"?", 0FFF7DED6, 2170},  
01247 {"?", 0FFF7DEDE, 2171},  
01248 {"?", 0FFE8418C, 2220},  
01249 {"?", 0FF8C0C4A, 2222},  
01250 {"?", 0FF883A40, 2224},
```

```
01251 {"?", 0xFFEE71A1, 2230},  
01252 {"?", 0xFFA95A68, 2241},  
01253 {"?", 0xFFFFAC8D3, 2250},  
01254 {"?", 0xFFD3007B, 2300},  
01255 {"?", 0xFFD20067, 2320},  
01256 {"?", 0xFFE651533, 2333},  
01257 {"?", 0xFF3A212B, 2336},  
01258 {"?", 0xFFFFDE5EC, 2363},  
01259 {"?", 0xFF970059, 2500},  
01260 {"?", 0xFFAA4381, 2504},  
01261 {"?", 0xFF820052, 2506},  
01262 {"?", 0xFFE20078, 2520},  
01263 {"?", 0xFFBF006A, 2521},  
01264 {"?", 0xFFF189AF, 2550},  
01265 {"?", 0xFFF7B4CA, 2560},  
01266 {"?", 0xFF494949, 2576},  
01267 {"?", 0xFF893480, 2600},  
01268 {"?", 0xFF6C0051, 2611},  
01269 {"?", 0xFFD994B9, 2640},  
01270 {"?", 0xFFE6B7CF, 2650},  
01271 {"?", 0xFFECD2DE, 2655},  
01272 {"?", 0xFF606D8C, 2674},  
01273 {"?", 0xFF610051, 2711},  
01274 {"?", 0xFF490251, 2715},  
01275 {"?", 0xFF89347F, 2720},  
01276 {"?", 0xFFC690A1, 2764},  
01277 {"?", 0xFF6F067B, 2810},  
01278 {"?", 0xFFA974AB, 2830},  
01279 {"?", 0xFF4C0F7B, 2900},  
01280 {"?", 0xFF664090, 2905},  
01281 {"?", 0xFF83589D, 2910},  
01282 {"?", 0xFF8C6DAA, 2920},  
01283 {"?", 0xFFC9B5D4, 3040},  
01284 {"?", 0xFFC790BA, 3045},  
01285 {"?", 0xFF707070, 3062},  
01286 {"?", 0xFF2A377E, 3102},  
01287 {"?", 0xFF35247A, 3110},  
01288 {"?", 0xFF260751, 3114},  
01289 {"?", 0xFF353A90, 3210},  
01290 {"?", 0xFF524A90, 3211},  
01291 {"?", 0xFF7D77AF, 3241},  
01292 {"?", 0xFF9083A3, 3251},  
01293 {"?", 0xFF14214E, 3323},  
01294 {"?", 0xFF7F8BC2, 3331},  
01295 {"?", 0xFF1B3C78, 3333},  
01296 {"?", 0xFF2E4B9B, 3335},  
01297 {"?", 0xFF11263C, 3344},  
01298 {"?", 0xFF202A65, 3353},  
01299 {"?", 0xFF171B4A, 3355},  
01300 {"?", 0xFF002232, 3444},  
01301 {"?", 0xFF2D4491, 3522},  
01302 {"?", 0xFF261257, 3536},  
01303 {"?", 0xFF3A2885, 3541},  
01304 {"?", 0xFF233B7D, 3543},  
01305 {"?", 0xFF273C82, 3544},  
01306 {"?", 0xFF272651, 3554},  
01307 {"?", 0xFF28438C, 3600},  
01308 {"?", 0xFF243A7D, 3611},  
01309 {"?", 0xFF4055A1, 3612},  
01310 {"?", 0xFF1A4C8D, 3622},  
01311 {"?", 0xFF92B1DC, 3640},  
01312 {"?", 0xFF648DC7, 3641},  
01313 {"?", 0xFFD0DDEE, 3650},  
01314 {"?", 0xFFC8D6ED, 3652},  
01315 {"?", 0xFF00507F, 3732},  
01316 {"?", 0xFFL2253C, 3743},  
01317 {"?", 0xFFB7D1E3, 3750},  
01318 {"?", 0xFFAFC9E5, 3761},  
01319 {"?", 0xFFCED2D1, 3770},  
01320 {"?", 0xFF3D6AA1, 3810},  
01321 {"?", 0xFF7BA2D3, 3815},  
01322 {"?", 0xFF91B9E2, 3820},  
01323 {"?", 0xFFB4CEE8, 3840},  
01324 {"?", 0xFF507193, 3842},  
01325 {"?", 0xFF007EBA, 3900},  
01326 {"?", 0xFF0082C4, 3901},  
01327 {"?", 0xFF006CA5, 3902},  
01328 {"?", 0xFF4ABDF0, 3910},  
01329 {"?", 0xFF86AAC8, 3951},  
01330 {"?", 0xFF697698, 3953},  
01331 {"?", 0xFFA6D8F6, 3962},  
01332 {"?", 0xFFE1E1E1, 3971},  
01333 {"?", 0xFF0093B9, 4010},  
01334 {"?", 0xFF507793, 4032},  
01335 {"?", 0xFF265674, 4033},  
01336 {"?", 0xFFEAFOF9, 4071},  
01337 {"?", 0xFF838689, 4073},
```

```

01338 {"?", 0xFF2DB0CF, 4101},
01339 {"?", 0xFF0095C6, 4103},
01340 {"?", 0xFF00A4D9, 4111},
01341 {"?", 0xFF00A9C9, 4113},
01342 {"?", 0xFF0082AD, 4116},
01343 {"?", 0xFF00405D, 4133},
01344 {"?", 0xFF192024, 4174}, /* TODO: duplicate case value */
01345 {"?", 0xFF4FB4CB, 4220},
01346 {"?", 0xFF8DCEE4, 4230},
01347 {"?", 0xFF8DCDDB, 4240},
01348 {"?", 0xFFD5EBF2, 4250},
01349 {"?", 0xFF007B8D, 4410},
01350 {"?", 0xFF0091A5, 4421},
01351 {"?", 0xFF007D8C, 4423},
01352 {"?", 0xFF007986, 4425},
01353 {"?", 0xFF5FBFD1, 4430},
01354 {"?", 0xFF006981, 4442},
01355 {"?", 0xFF007F92, 4452},
01356 {"?", 0xFF002F38, 4515},
01357 {"?", 0xFF007389, 4531},
01358 {"?", 0xFF007B8D, 4610}, /* TODO: duplicate case value */
01359 {"?", 0xFF00A3A0, 4620},
01360 {"?", 0xFF0B7F85, 4625},
01361 {"?", 0xFF005B63, 4643},
01362 {"?", 0xFF234544, 4644},
01363 {"?", 0xFF005B63, 5005}, /* TODO: duplicate case value */
01364 {"?", 0xFF00A6AD, 5010},
01365 {"?", 0xFFB4DCD8, 5050},
01366 {"?", 0xFF00876E, 5100},
01367 {"?", 0xFF009084, 5101},
01368 {"?", 0xFF48BAB7, 5115},
01369 {"?", 0xFF00AF8C, 5210},
01370 {"?", 0xFF8CCCC2, 5220},
01371 {"?", 0xFF47B9AE, 5230},
01372 {"?", 0xFF197E6D, 5233},
01373 {"?", 0xFF006E42, 5324},
01374 {"?", 0xFF004D3D, 5326},
01375 {"?", 0xFF002F38, 5335}, /* TODO: duplicate case value */
01376 {"?", 0xFF002D1F, 5374},
01377 {"?", 0xFF008663, 5411},
01378 {"?", 0xFF006B4E, 5415},
01379 {"?", 0xFF008663, 5422}, /* TODO: duplicate case value */
01380 {"?", 0xFF52BA84, 5500},
01381 {"?", 0xFF14A363, 5510},
01382 {"?", 0xFF007848, 5513},
01383 {"?", 0xFF008663, 5515}, /* TODO: duplicate case value */
01384 {"?", 0xFF52A04F, 5531},
01385 {"?", 0xFF94ADA1, 5552},
01386 {"?", 0xFF103828, 5555},
01387 {"?", 0xFF85C875, 5610},
01388 {"?", 0xFF14B26D, 5613},
01389 {"?", 0xFF1A781E, 5633},
01390 {"?", 0xFF157436, 5643},
01391 {"?", 0xFFC9E3C5, 5650},
01392 {"?", 0xFF6B9181, 5664},
01393 {"?", 0xFFA5C278, 5822},
01394 {"?", 0xFF70953F, 5833},
01395 {"?", 0xFF273014, 5866},
01396 {"?", 0xFF81C750, 5912},
01397 {"?", 0xFF457021, 5933},
01398 {"?", 0xFF506702, 5934},
01399 {"?", 0xFFBBD841, 5940},
01400 {"?", 0xFF003518, 5944},
01401 {"?", 0FFE3EB00, 6010},
01402 {"?", 0xFFBED782, 6051},
01403 {"?", 0xFF919600, 6133},
01404 {"?", 0xFF484601, 6156},
01405 {"END", 0, -1}
01406 };
01407
01408 thread_color Isafil_Rayon_codes[] = {
01409 {"?", 0xFFFFFFFF, 101},
01410 {"?", 0xFFFFFFFF, 15}, /* TODO: duplicate case value */
01411 {"?", 0xFFFFFFFF, 17}, /* TODO: duplicate case value */
01412 {"?", 0xFFFF000000, 20},
01413 {"?", 0xFFFFFDDED, 101},
01414 {"?", 0xFF7D7D7D, 104},
01415 {"?", 0xFF515B61, 107},
01416 {"?", 0xFF6D757B, 108},
01417 {"?", 0xFFACACAC, 109},
01418 {"?", 0xFF515B61, 111}, /* TODO: duplicate case value */
01419 {"?", 0xFF5D5D5D, 112},
01420 {"?", 0xFFFCFCFC, 124},
01421 {"?", 0xFFA1A9B4, 131},
01422 {"?", 0xFF6D757B, 141}, /* TODO: duplicate case value */
01423 {"?", 0xFF9EA5AA, 142},
01424 {"?", 0xFFCFD1D5, 145},

```

```
01425 {"?", 0xFFC6BDB4, 150},  
01426 {"?", 0xFFD5C4B3, 151},  
01427 {"?", 0xFF7C8283, 152},  
01428 {"?", 0xFF898F94, 156},  
01429 {"?", 0xFFFFEF5F0, 180},  
01430 {"?", 0xFFE9D7D9, 182},  
01431 {"?", 0xFFEBE3DD, 184},  
01432 {"?", 0xFFE0DA5F, 221},  
01433 {"?", 0xFFFBFA28, 232},  
01434 {"?", 0xFFCE9C1, 241},  
01435 {"?", 0xFFFFAF6CC, 250},  
01436 {"?", 0xFFCE7A5, 251},  
01437 {"?", 0xFFCEADB, 260},  
01438 {"?", 0xFFF9F8E8, 270},  
01439 {"?", 0xFFFFDF76C, 310},  
01440 {"?", 0xFFF5D300, 311},  
01441 {"?", 0xFF797E24, 345},  
01442 {"?", 0xFFB0AA76, 352},  
01443 {"?", 0xFF898F2B, 442},  
01444 {"?", 0xFF98996D, 453},  
01445 {"?", 0xFF6E772E, 454},  
01446 {"?", 0xFF6B7E6F, 463},  
01447 {"?", 0xFF3E4F34, 465},  
01448 {"?", 0xFFEDEF05, 501},  
01449 {"?", 0xFFFFAF6CC, 505}, /* TODO: duplicate case value */  
01450 {"?", 0xFFF5D300, 506}, /* TODO: duplicate case value */  
01451 {"?", 0xFFFFFB1, 510},  
01452 {"?", 0xFFFFDE896, 520},  
01453 {"?", 0xFFD7CE8A, 532},  
01454 {"?", 0xFFB18B00, 542},  
01455 {"?", 0xFFAA8D00, 545},  
01456 {"?", 0xFFB28F11, 546},  
01457 {"?", 0xFFAC9436, 551},  
01458 {"?", 0xFFB69F56, 552},  
01459 {"?", 0xFFF4EE8C, 580},  
01460 {"?", 0xFFF1EB35, 590},  
01461 {"?", 0xFFF8D73E, 600},  
01462 {"?", 0xFFF8D73E, 605}, /* TODO: duplicate case value */  
01463 {"?", 0xFFF7CB47, 610},  
01464 {"?", 0xFFF7E400, 615},  
01465 {"?", 0xFFFFDE896, 620}, /* TODO: duplicate case value */  
01466 {"?", 0xFFEEDB00, 625},  
01467 {"?", 0xFFFFF09A, 630},  
01468 {"?", 0xFFFFDE1AF, 635},  
01469 {"?", 0xFFF5D2A6, 651},  
01470 {"?", 0xFFFFF9EA, 660},  
01471 {"?", 0xFFFFAF6E7, 670},  
01472 {"?", 0xFFEBE8A8, 672},  
01473 {"?", 0xFFF7C35F, 700},  
01474 {"?", 0xFFF5CA00, 702},  
01475 {"?", 0xFFDFA200, 704},  
01476 {"?", 0xFFFFCF538, 706},  
01477 {"?", 0xFFFFADC59, 713},  
01478 {"?", 0xFF8C7E6A, 722},  
01479 {"?", 0xFF827000, 726},  
01480 {"?", 0xFF636254, 732},  
01481 {"?", 0xFF594900, 747},  
01482 {"?", 0xFFD6BF94, 761},  
01483 {"?", 0xFF656452, 776},  
01484 {"?", 0xFFF1AF00, 800},  
01485 {"?", 0xFFF3C200, 805},  
01486 {"?", 0xFFF5BA5D, 811},  
01487 {"?", 0xFFE1A23B, 821},  
01488 {"?", 0xFFCCAB3F, 822},  
01489 {"?", 0xFFFFFA200, 824}, /* TODO: duplicate case value */  
01490 {"?", 0xFFF3B044, 830},  
01491 {"?", 0xFFDOA44F, 832},  
01492 {"?", 0xFFCD944A, 842},  
01493 {"?", 0xFFE3BC61, 851},  
01494 {"?", 0xFF947C4A, 853},  
01495 {"?", 0xFF001F48, 866},  
01496 {"?", 0xFFEBE7DD, 870},  
01497 {"?", 0xFF9FA086, 873},  
01498 {"?", 0xFF9A897B, 874},  
01499 {"?", 0xFFEE9C00, 900},  
01500 {"?", 0xFF3B259, 904},  
01501 {"?", 0xFFCA832C, 922},  
01502 {"?", 0xFFC07314, 931},  
01503 {"?", 0xFFAC6613, 932},  
01504 {"?", 0xFF744808, 933},  
01505 {"?", 0xFFBD9565, 934},  
01506 {"?", 0xFF806800, 936},  
01507 {"?", 0xFFC98300, 940},  
01508 {"?", 0xFFFFAF7D3E, 941},  
01509 {"?", 0xFF483928, 945},  
01510 {"?", 0xFFFFECD9, 961},  
01511 {"?", 0xFFFFECD9, 961},
```

```
01512 {"?", 0xFFFFEFEED, 970},  
01513 {"?", 0xFFDD973A, 1041},  
01514 {"?", 0xFF6A4129, 1055},  
01515 {"?", 0xFFFFDE2C1, 1060},  
01516 {"?", 0xFFA68A68, 1061},  
01517 {"?", 0xFFD76814, 1099},  
01518 {"?", 0xFFED873F, 1100},  
01519 {"?", 0xFFEC870E, 1101},  
01520 {"?", 0xFFD9206, 1102},  
01521 {"?", 0xFFEE9C00, 1106}, /* TODO: duplicate case value */  
01522 {"?", 0xFFC45331, 1113},  
01523 {"?", 0xFFEE8751, 1114},  
01524 {"?", 0xFFA35214, 1115},  
01525 {"?", 0xFFFF8C000, 1120},  
01526 {"?", 0xFFB7976B, 1123},  
01527 {"?", 0xFFD95A21, 1134},  
01528 {"?", 0xFFF3D8A8, 1140},  
01529 {"?", 0xFFFFACFAE, 1141},  
01530 {"?", 0xFFDFC8AB, 1172},  
01531 {"?", 0xFFE89763, 1211},  
01532 {"?", 0xFFF1A236, 1220},  
01533 {"?", 0xFF3D2723, 1276},  
01534 {"?", 0FFE5571D, 1300},  
01535 {"?", 0xFFE8643C, 1302},  
01536 {"?", 0FFE4501E, 1304},  
01537 {"?", 0FFEA7134, 1305},  
01538 {"?", 0FFE12A23, 1306},  
01539 {"?", 0FFC14817, 1311},  
01540 {"?", 0FFC45331, 1312}, /* TODO: duplicate case value */  
01541 {"?", 0FFD7623E, 1313},  
01542 {"?", 0FFED7C56, 1320},  
01543 {"?", 0FF92291B, 1324},  
01544 {"?", 0FFD5815E, 1332},  
01545 {"?", 0FFBB3D2E, 1334},  
01546 {"?", 0FFBE2D1A, 1335},  
01547 {"?", 0FF5F1B23, 1342},  
01548 {"?", 0FFT7A3441, 1346},  
01549 {"?", 0FF84291D, 1348},  
01550 {"?", 0FFFBBF95, 1351},  
01551 {"?", 0FFF4A773, 1352},  
01552 {"?", 0FF693920, 1355},  
01553 {"?", 0FFF9C6A1, 1361},  
01554 {"?", 0FFF9C598, 1362},  
01555 {"?", 0FF432731, 1366},  
01556 {"?", 0FF464537, 1375},  
01557 {"?", 0FF4D2E18, 1376},  
01558 {"?", 0FFD64F42, 1421},  
01559 {"?", 0FF4A782, 1430},  
01560 {"?", 0FFE22D2A, 1501},  
01561 {"?", 0FFA93121, 1514},  
01562 {"?", 0FFEC7168, 1521},  
01563 {"?", 0FF6B08E, 1532},  
01564 {"?", 0FFF9C5B9, 1551},  
01565 {"?", 0FF806A61, 1565},  
01566 {"?", 0FF464537, 1573}, /* TODO: duplicate case value */  
01567 {"?", 0FFE36C63, 1600},  
01568 {"?", 0FFF9C7B9, 1630},  
01569 {"?", 0FFE44733, 1701},  
01570 {"?", 0FFDF0032, 1703},  
01571 {"?", 0FFE44733, 1705}, /* TODO: duplicate case value */  
01572 {"?", 0FFCF0040, 1725},  
01573 {"?", 0FFDB686B, 1750},  
01574 {"?", 0FFF1CDCE, 1755},  
01575 {"?", 0FFE9C9BD, 1760},  
01576 {"?", 0FFEB8C0B8, 1761},  
01577 {"?", 0FFE00046, 1800},  
01578 {"?", 0FFE43449, 1802},  
01579 {"?", 0FFD6445D, 1805},  
01580 {"?", 0FFF49E95, 1840},  
01581 {"?", 0FFB76663, 1842},  
01582 {"?", 0FFE36C63, 1849}, /* TODO: duplicate case value */  
01583 {"?", 0FFF0887D, 1850},  
01584 {"?", 0FFFACTC1, 1855},  
01585 {"?", 0FFFCDA5, 1860},  
01586 {"?", 0FFFDE3D3, 1870},  
01587 {"?", 0FFG36254, 1874}, /* TODO: duplicate case value */  
01588 {"?", 0FF394535, 1876},  
01589 {"?", 0FFE10057, 1900},  
01590 {"?", 0FFBD0041, 1902},  
01591 {"?", 0FFC00343, 1903},  
01592 {"?", 0FFA9023A, 1904},  
01593 {"?", 0FF960018, 1905},  
01594 {"?", 0FFBE004F, 1906},  
01595 {"?", 0FF910230, 1911},  
01596 {"?", 0FF86023E, 1912},  
01597 {"?", 0FF9A0C3B, 1913},  
01598 {"?", 0FFA41F39, 1914},
```

```
01599 {"?", 0xFFA33050, 1921},  
01600 {"?", 0xFFFF28DA6, 1940},  
01601 {"?", 0xFFCE427A, 1950},  
01602 {"?", 0xFFF2B9BE, 1960},  
01603 {"?", 0xFF959595, 1972},  
01604 {"?", 0xFFA33145, 2011},  
01605 {"?", 0xFF9F454C, 2022},  
01606 {"?", 0xFFC7979B, 2051},  
01607 {"?", 0xFFD18D89, 2053},  
01608 {"?", 0xFF970038, 2100},  
01609 {"?", 0xFF9F003F, 2101},  
01610 {"?", 0xFF78093F, 2113},  
01611 {"?", 0xFF432732, 2123},  
01612 {"?", 0xFFE6778B, 2152},  
01613 {"?", 0xFFDF8390, 2153},  
01614 {"?", 0xFFF9BFC0, 2155},  
01615 {"?", 0xFFFFBD1D6, 2160},  
01616 {"?", 0xFFFFDE3DB, 2165},  
01617 {"?", 0xFFD8D5D0, 2166},  
01618 {"?", 0xFFFFEDE2, 2168},  
01619 {"?", 0xFFF7DED6, 2170},  
01620 {"?", 0xFFF7DEDE, 2171},  
01621 {"?", 0xFFFFCD9C4, 2180},  
01622 {"?", 0xFFE8418C, 2220},  
01623 {"?", 0xFF8C0C4A, 2222},  
01624 {"?", 0xFF883A40, 2224},  
01625 {"?", 0xFFEE71A1, 2230},  
01626 {"?", 0xFFA95A68, 2241},  
01627 {"?", 0xFFFFAC8D3, 2250},  
01628 {"?", 0xFFD3007E, 2300},  
01629 {"?", 0xFFBF006A, 2310},  
01630 {"?", 0xFFD20067, 2320},  
01631 {"?", 0xFF780C38, 2332},  
01632 {"?", 0xFF651533, 2333},  
01633 {"?", 0xFF3A212B, 2336},  
01634 {"?", 0xFFF2E0DC, 2360},  
01635 {"?", 0xFFFFDE5EC, 2363},  
01636 {"?", 0xFF970059, 2500},  
01637 {"?", 0xFF8B1771, 2502},  
01638 {"?", 0xFFAA4381, 2504},  
01639 {"?", 0xFFB40073, 2505},  
01640 {"?", 0xFF820052, 2506},  
01641 {"?", 0xFFD63C81, 2513},  
01642 {"?", 0FFE20078, 2520},  
01643 {"?", 0xFFBF006A, 2521}, /* TODO: duplicate case value */  
01644 {"?", 0xFFE71A1, 2524}, /* TODO: duplicate case value */  
01645 {"?", 0xFFF189AF, 2550},  
01646 {"?", 0FFF7B4CA, 2555},  
01647 {"?", 0FFF7B4CA, 2560}, /* TODO: duplicate case value */  
01648 {"?", 0xFF494949, 2576},  
01649 {"?", 0xFF394248, 2578},  
01650 {"?", 0xFF893480, 2600},  
01651 {"?", 0xFF6C0051, 2611},  
01652 {"?", 0xFFCD73A6, 2620},  
01653 {"?", 0FFD994B9, 2640},  
01654 {"?", 0FFDDBDD5, 2645},  
01655 {"?", 0FFE6B7CF, 2650},  
01656 {"?", 0FFECD2DE, 2655},  
01657 {"?", 0FF606D8C, 2674},  
01658 {"?", 0FF646A6B, 2675},  
01659 {"?", 0FF610051, 2711},  
01660 {"?", 0FFT04191, 2712},  
01661 {"?", 0FF490251, 2715},  
01662 {"?", 0FF2F206F, 2743},  
01663 {"?", 0FFC690A1, 2764},  
01664 {"?", 0FF6F067B, 2810},  
01665 {"?", 0FFAD85B1, 2820},  
01666 {"?", 0FFA974AB, 2830},  
01667 {"?", 0FF4C0F7B, 2900},  
01668 {"?", 0FF664090, 2905},  
01669 {"?", 0FF83589D, 2910},  
01670 {"?", 0FF8C6DAA, 2920},  
01671 {"?", 0FFC9B5D4, 3040},  
01672 {"?", 0FFC790BA, 3045},  
01673 {"?", 0FF707070, 3062},  
01674 {"?", 0FF2A377B, 3102},  
01675 {"?", 0FF3C1F81, 3103},  
01676 {"?", 0FF35247A, 3110},  
01677 {"?", 0FF260751, 3114},  
01678 {"?", 0FF28135B, 3133},  
01679 {"?", 0FF6E5DA3, 3200},  
01680 {"?", 0FF353A90, 3210},  
01681 {"?", 0FF524A90, 3211},  
01682 {"?", 0FF785FA3, 3213},  
01683 {"?", 0FF241757, 3222},  
01684 {"?", 0FF7D77AF, 3241},  
01685 {"?", 0FF9083A3, 3251},
```

```
01686 {"?", 0xFFB2AABD, 3262},  
01687 {"?", 0xFF392D88, 3301},  
01688 {"?", 0xFF5661A8, 3321},  
01689 {"?", 0xFF323887, 3322},  
01690 {"?", 0xFF14214E, 3323},  
01691 {"?", 0xFF3A2885, 3330},  
01692 {"?", 0xFF7F8BC2, 3331},  
01693 {"?", 0xFF1B3C78, 3333},  
01694 {"?", 0xFFB9BDD9, 3340},  
01695 {"?", 0xFF11263C, 3344},  
01696 {"?", 0xFF202A65, 3353},  
01697 {"?", 0xFF171B4A, 3355},  
01698 {"?", 0xFF959ACA, 3420},  
01699 {"?", 0xFF6A76B5, 3430},  
01700 {"?", 0xFF11263C, 3443}, /* TODO: duplicate case value */  
01701 {"?", 0xFF002232, 3444},  
01702 {"?", 0xFF2D4491, 3522},  
01703 {"?", 0xFF261257, 3536},  
01704 {"?", 0xFF53428A, 3540},  
01705 {"?", 0xFF3A2885, 3541}, /* TODO: duplicate case value */  
01706 {"?", 0xFF233B7D, 3543},  
01707 {"?", 0xFF273C82, 3544},  
01708 {"?", 0xFF272651, 3554},  
01709 {"?", 0xFF28438C, 3600},  
01710 {"?", 0xFF243A7D, 3611},  
01711 {"?", 0xFF4055A1, 3612},  
01712 {"?", 0xFF1A4C8D, 3622},  
01713 {"?", 0xFF1E569F, 3631},  
01714 {"?", 0xFF92B1DC, 3640},  
01715 {"?", 0xFF648DC7, 3641},  
01716 {"?", 0xFFD0DEEE, 3650},  
01717 {"?", 0xFFEAFOF9, 3661},  
01718 {"?", 0xFF00507E, 3732},  
01719 {"?", 0xFF12253C, 3743},  
01720 {"?", 0xFFB7D1E3, 3750},  
01721 {"?", 0xFFD0DEEE, 3752}, /* TODO: duplicate case value */  
01722 {"?", 0xFFAFC9E5, 3761},  
01723 {"?", 0xFFCED2D1, 3770},  
01724 {"?", 0xFF3D6AA1, 3810},  
01725 {"?", 0xFF91B9E2, 3820},  
01726 {"?", 0xFF00779E, 3822},  
01727 {"?", 0xFFB4CEEB, 3840},  
01728 {"?", 0xFF507193, 3842},  
01729 {"?", 0xFFD5E3F4, 3845},  
01730 {"?", 0xFF9AB8D3, 3851},  
01731 {"?", 0xFF007EBA, 3900},  
01732 {"?", 0xFF0082C4, 3901},  
01733 {"?", 0xFF006CA5, 3902},  
01734 {"?", 0xFF4ABDF0, 3910},  
01735 {"?", 0xFF86AAC, 3951},  
01736 {"?", 0xFF485E8A, 3952},  
01737 {"?", 0xFF697698, 3953},  
01738 {"?", 0xFFC5E1F3, 3961},  
01739 {"?", 0xFFA6D8F6, 3962},  
01740 {"?", 0xFFE1E1E1, 3971},  
01741 {"?", 0xFF0093B9, 4010},  
01742 {"?", 0xFF006587, 4022},  
01743 {"?", 0xFF87C7EA, 4030},  
01744 {"?", 0xFF507793, 4032},  
01745 {"?", 0xFF265674, 4033},  
01746 {"?", 0xFF9ED4E6, 4040},  
01747 {"?", 0xFFEAFOF9, 4071}, /* TODO: duplicate case value */  
01748 {"?", 0xFF0096C1, 4100},  
01749 {"?", 0xFF2DB0CF, 4101},  
01750 {"?", 0xFF0095C6, 4103},  
01751 {"?", 0xFF0081AA, 4105},  
01752 {"?", 0xFF00A4D9, 4111},  
01753 {"?", 0xFF00A9C9, 4113},  
01754 {"?", 0xFF5DBFD2, 4121},  
01755 {"?", 0xFF00405D, 4133},  
01756 {"?", 0xFF192024, 4174}, /* TODO: duplicate case value */  
01757 {"?", 0xFF192024, 4175}, /* TODO: duplicate case value */  
01758 {"?", 0xFF4FB4CB, 4220},  
01759 {"?", 0xFF8DCEE4, 4230},  
01760 {"?", 0xFF2DB0CF, 4231}, /* TODO: duplicate case value */  
01761 {"?", 0xFF006587, 4232}, /* TODO: duplicate case value */  
01762 {"?", 0xFF8DCDDB, 4240},  
01763 {"?", 0xFFD5EBF2, 4250},  
01764 {"?", 0xFF007389, 4400},  
01765 {"?", 0xFF007B8D, 4410},  
01766 {"?", 0xFF00B2CA, 4420},  
01767 {"?", 0xFF0091A5, 4421},  
01768 {"?", 0xFF007D8C, 4423},  
01769 {"?", 0xFF007986, 4425},  
01770 {"?", 0xFF5FBFD1, 4430},  
01771 {"?", 0xFF004250, 4432},  
01772 {"?", 0xFF8DCEE4, 4440}, /* TODO: duplicate case value */
```

```
01773 {"?", 0xFF006981, 4442},  
01774 {"?", 0xFF007F92, 4452},  
01775 {"?", 0xFF008192, 4500},  
01776 {"?", 0xFF007079, 4513},  
01777 {"?", 0xFF002F38, 4515},  
01778 {"?", 0xFF00646A, 4516},  
01779 {"?", 0xFF007389, 4531}, /* TODO: duplicate case value */  
01780 {"?", 0xFF007B8D, 4610}, /* TODO: duplicate case value */  
01781 {"?", 0xFF00A3A0, 4620},  
01782 {"?", 0xFF0B7F85, 4625},  
01783 {"?", 0xFF005B63, 4643},  
01784 {"?", 0xFF234544, 4644},  
01785 {"?", 0xFF8CCDD3, 4840},  
01786 {"?", 0xFF006F73, 5000},  
01787 {"?", 0xFF005B63, 5005}, /* TODO: duplicate case value */  
01788 {"?", 0xFF00A6AD, 5010},  
01789 {"?", 0xFF49BAC0, 5020},  
01790 {"?", 0xFFCFDDE0, 5040},  
01791 {"?", 0xFFE4DCD8, 5050},  
01792 {"?", 0xFF00876E, 5100},  
01793 {"?", 0xFF009084, 5101},  
01794 {"?", 0xFF00B1AE, 5111},  
01795 {"?", 0xFF48BAB7, 5115},  
01796 {"?", 0xFF00AF8C, 5210},  
01797 {"?", 0xFF8CCCC2, 5220},  
01798 {"?", 0xFF47B9AE, 5230},  
01799 {"?", 0xFF197E6D, 5233},  
01800 {"?", 0xFF8CCCC2, 5240}, /* TODO: duplicate case value */  
01801 {"?", 0xFF005B63, 5255}, /* TODO: duplicate case value */  
01802 {"?", 0xFF006E42, 5324},  
01803 {"?", 0xFF004D3D, 5326},  
01804 {"?", 0xFF002F38, 5335}, /* TODO: duplicate case value */  
01805 {"?", 0xFF002D1F, 5374},  
01806 {"?", 0xFF002F38, 5375}, /* TODO: duplicate case value */  
01807 {"?", 0xFF008663, 5411},  
01808 {"?", 0xFF006B4E, 5415},  
01809 {"?", 0xFF008663, 5422}, /* TODO: duplicate case value */  
01810 {"?", 0xFF006B56, 5425},  
01811 {"?", 0xFF008879, 5431},  
01812 {"?", 0xFFDBE9E5, 5460},  
01813 {"?", 0xFF6AC093, 5470},  
01814 {"?", 0xFF52BA84, 5500},  
01815 {"?", 0xFF14A363, 5510},  
01816 {"?", 0xFF007848, 5513},  
01817 {"?", 0xFF008663, 5515}, /* TODO: duplicate case value */  
01818 {"?", 0xFF52A04F, 5531},  
01819 {"?", 0xFF6EA293, 5542},  
01820 {"?", 0xFF94ADA1, 5552},  
01821 {"?", 0xFF103828, 5555},  
01822 {"?", 0xFF63BE5F, 5600},  
01823 {"?", 0xFF85C875, 5610},  
01824 {"?", 0xFF2CB431, 5611},  
01825 {"?", 0xFF14B26D, 5613},  
01826 {"?", 0xFF40B780, 5620},  
01827 {"?", 0xFF1A781E, 5633},  
01828 {"?", 0xFF157436, 5643},  
01829 {"?", 0xFFC9E3C5, 5650},  
01830 {"?", 0xFF6B9181, 5664},  
01831 {"?", 0xFF3A6D57, 5765},  
01832 {"?", 0xFF103828, 5766}, /* TODO: duplicate case value */  
01833 {"?", 0xFF02140C, 5776},  
01834 {"?", 0xFFA5C278, 5822},  
01835 {"?", 0xFFB4D383, 5832},  
01836 {"?", 0xFF70953F, 5833},  
01837 {"?", 0xFFA2D289, 5840},  
01838 {"?", 0xFF273014, 5866},  
01839 {"?", 0xFF81C750, 5912},  
01840 {"?", 0xFF457021, 5933},  
01841 {"?", 0xFF506702, 5934},  
01842 {"?", 0xFFBBDB41, 5940},  
01843 {"?", 0xFF003518, 5944},  
01844 {"?", 0FFE3EB00, 6010},  
01845 {"?", 0xFFBED782, 6051},  
01846 {"?", 0xFF2D3B01, 6065},  
01847 {"?", 0xFFDCDDD1, 6071},  
01848 {"?", 0xFF919600, 6133},  
01849 {"?", 0xFF484601, 6156},  
01850 {"END", 0, -1}  
01851 };  
01852  
01853 thread_color Marathon_Polyester_codes[] = {  
01854 {"END", 0, -1}  
01855 };  
01856  
01857 thread_color Marathon_Rayon_codes[] = {  
01858 {"END", 0, -1}  
01859 };
```

```
01860
01861     thread_color Madeira_Polyester_codes[] = {
01862         {"END", 0, -1}
01863     };
01864
01865     thread_color Madeira_Rayon_codes[] = {
01866         {"END", 0, -1}
01867     };
01868
01869     thread_color Metro_Polyester_codes[] = {
01870         {"END", 0, -1}
01871     };
01872
01873     thread_color Pantone_codes[] = {
01874         {"?", 0xFFFFF7D, 100},
01875         {"?", 0xFFFFF36, 101},
01876         {"?", 0xFFFFFC0D, 102},
01877         {"?", 0xFFD1CB00, 103},
01878         {"?", 0xFFB3AD00, 104},
01879         {"?", 0xFF807C00, 105},
01880         {"?", 0xFFFFFA4F, 106},
01881         {"?", 0xFFFFF536, 107},
01882         {"?", 0xFFFFF00D, 108},
01883         {"?", 0xFFFFE600, 109},
01884         {"?", 0xFFEDD100, 110},
01885         {"?", 0xFFBA600, 111},
01886         {"?", 0xFF9E8E00, 112},
01887         {"?", 0xFFFFED57, 113},
01888         {"?", 0xFFFFEB45, 114},
01889         {"?", 0xFFFFE833, 115},
01890         {"?", 0xFFFFD600, 116},
01891         {"?", 0xFFD9B200, 117},
01892         {"?", 0xFFBA9900, 118},
01893         {"?", 0xFF827200, 119},
01894         {"?", 0xFFFFE86B, 120},
01895         {"?", 0xFFFFF2B0, 1205},
01896         {"?", 0xFFFFE34F, 121},
01897         {"?", 0xFFFFE88C, 1215},
01898         {"?", 0xFFFFD433, 122},
01899         {"?", 0xFFFFD461, 1225},
01900         {"?", 0xFFFFC20F, 123},
01901         {"?", 0xFFFFB517, 1235},
01902         {"?", 0xFFFF0AD00, 124},
01903         {"?", 0xFFD19700, 1245},
01904         {"?", 0xFFBD8C00, 125},
01905         {"?", 0xFFA87B00, 1255},
01906         {"?", 0xFFA17800, 126},
01907         {"?", 0xFF7D5B00, 1265},
01908         {"?", 0xFFFFED80, 127},
01909         {"?", 0xFFFFE359, 128},
01910         {"?", 0xFFFFD63B, 129},
01911         {"?", 0xFFFFB300, 130},
01912         {"?", 0xFFE89E00, 131},
01913         {"?", 0xFFB38100, 132},
01914         {"?", 0xFF705A00, 133},
01915         {"?", 0xFFFFE38C, 134},
01916         {"?", 0xFFFFDB87, 1345},
01917         {"?", 0xFFFFCF66, 135},
01918         {"?", 0xFFFFCC70, 1355},
01919         {"?", 0xFFFFBA3D, 136},
01920         {"?", 0xFFFFB547, 1365},
01921         {"?", 0xFFFFA61A, 137},
01922         {"?", 0xFFFF991A, 1375},
01923         {"?", 0xFFFFC9200, 138},
01924         {"?", 0xFFED8500, 1385},
01925         {"?", 0xFFC47C00, 139},
01926         {"?", 0xFFA15F00, 1395},
01927         {"?", 0xFF755600, 140},
01928         {"?", 0xFF5E3C00, 1405},
01929         {"?", 0xFFFFCF7D, 141},
01930         {"?", 0xFFFFB83D, 142},
01931         {"?", 0xFFFFA626, 143},
01932         {"?", 0xFFFF8500, 144},
01933         {"?", 0xFFEB7C00, 145},
01934         {"?", 0xFFAB6100, 146},
01935         {"?", 0xFF705100, 147},
01936         {"?", 0xFFFFD6A1, 148},
01937         {"?", 0xFFFFBA75, 1485},
01938         {"?", 0xFFFFCF487, 149},
01939         {"?", 0xFFFFFAB54, 1495},
01940         {"?", 0xFFFFFA64D, 150},
01941         {"?", 0xFFFF943B, 1505},
01942         {"?", 0xFFFF850D, 151},
01943         {"?", 0xFFFFC7C00, 152},
01944         {"?", 0FFE66000, 1525},
01945         {"?", 0FFD17100, 153},
01946         {"?", 0FF9E4A00, 1535},
```

```
01947 {"?", 0xFFA85B00, 154},  
01948 {"?", 0xFFFF72200, 1545},  
01949 {"?", 0xFFFFE0B8, 155},  
01950 {"?", 0xFFFFFC7A8, 1555},  
01951 {"?", 0xFFFFFC794, 156},  
01952 {"?", 0xFFFFFA882, 1565},  
01953 {"?", 0xFFFF914D, 157},  
01954 {"?", 0xFFFF8C47, 1575},  
01955 {"?", 0xFFFF6308, 158},  
01956 {"?", 0xFFFF701A, 1585},  
01957 {"?", 0xFFED5100, 159},  
01958 {"?", 0xFFFF26300, 1595},  
01959 {"?", 0xFFAD4200, 160},  
01960 {"?", 0xFFB34F00, 1605},  
01961 {"?", 0xFF5C2C00, 161},  
01962 {"?", 0xFF914000, 1615},  
01963 {"?", 0xFFFFD9C7, 162},  
01964 {"?", 0xFFFFB0A1, 1625},  
01965 {"?", 0xFFFFB08F, 163},  
01966 {"?", 0xFFFF9C85, 1635},  
01967 {"?", 0xFFFF8A45, 164},  
01968 {"?", 0xFFFF8257, 1645},  
01969 {"?", 0xFFFF690A, 165},  
01970 {"?", 0xFFFF5E17, 1655},  
01971 {"?", 0xFFFF5C00, 166},  
01972 {"?", 0xFFFF5200, 1665},  
01973 {"?", 0xFFD45500, 167},  
01974 {"?", 0xFFB83D00, 1675},  
01975 {"?", 0xFF692D00, 168},  
01976 {"?", 0xFF8F2E00, 1685},  
01977 {"?", 0xFFFFCCCC, 169},  
01978 {"?", 0xFFFF998F, 170},  
01979 {"?", 0xFFFF7852, 171},  
01980 {"?", 0xFFFF571F, 172},  
01981 {"?", 0xFFFF54C00, 173},  
01982 {"?", 0xFFA33100, 174},  
01983 {"?", 0xFF662400, 175},  
01984 {"?", 0xFFFFBF01, 176},  
01985 {"?", 0xFFFF9EC9, 1765},  
01986 {"?", 0xFFFFBAE0, 1767},  
01987 {"?", 0xFFFF8C99, 177},  
01988 {"?", 0xFFFF87B5, 1775},  
01989 {"?", 0xFFFF6BA3, 1777},  
01990 {"?", 0xFFFF6970, 178},  
01991 {"?", 0xFFFF5480, 1785},  
01992 {"?", 0xFFFF3D66, 1787},  
01993 {"?", 0xFFFF291F, 1788},  
01994 {"?", 0xFFFF3600, 179},  
01995 {"?", 0xFFFF0F00, 1795},  
01996 {"?", 0xFFF50002, 1797},  
01997 {"?", 0xFFE33000, 180},  
01998 {"?", 0xFFC41200, 1805},  
01999 {"?", 0xFFB80007, 1807},  
02000 {"?", 0xFF872300, 181},  
02001 {"?", 0xFF7D0C00, 1815},  
02002 {"?", 0xFF570900, 1817},  
02003 {"?", 0xFFFFBDE6, 182},  
02004 {"?", 0xFFFF8AC9, 183},  
02005 {"?", 0xFFFF5296, 184},  
02006 {"?", 0xFFFF173D, 185},  
02007 {"?", 0xFFF5002F, 186},  
02008 {"?", 0xFFCC002B, 187},  
02009 {"?", 0xFF800400, 188},  
02010 {"?", 0xFFFFA1E6, 189},  
02011 {"?", 0xFFFFB8ED, 1895},  
02012 {"?", 0xFFFF73C7, 190},  
02013 {"?", 0xFFFF96E8, 1905},  
02014 {"?", 0xFFFF3D9E, 191},  
02015 {"?", 0xFFFF4ACC, 1915},  
02016 {"?", 0xFFFF0052, 192},  
02017 {"?", 0xFFFF0073, 1925},  
02018 {"?", 0xFFDE004B, 193},  
02019 {"?", 0xFFF20068, 1935},  
02020 {"?", 0xFFA8003E, 194},  
02021 {"?", 0xFFCF005B, 1945},  
02022 {"?", 0xFF73002B, 195},  
02023 {"?", 0xFFA10040, 1955},  
02024 {"?", 0xFFFFBFF5, 196},  
02025 {"?", 0xFFFF8CE6, 197},  
02026 {"?", 0xFFFF38AB, 198},  
02027 {"?", 0xFFFF0061, 199},  
02028 {"?", 0FFE00053, 200},  
02029 {"?", 0xFFB50043, 201},  
02030 {"?", 0xFF910039, 202},  
02031 {"?", 0xFFFFA8F7, 203},  
02032 {"?", 0xFFFF6BF7, 204},  
02033 {"?", 0xFFFF29E8, 205},
```

```
02034 {"?", 0xFFFF70099, 206},  
02035 {"?", 0xFFCF0076, 207},  
02036 {"?", 0xFFA10067, 208},  
02037 {"?", 0xFF78004F, 209},  
02038 {"?", 0xFFFF9CFO, 210},  
02039 {"?", 0xFFFF73EB, 211},  
02040 {"?", 0xFFFF47E3, 212},  
02041 {"?", 0xFFFF0DBA, 213},  
02042 {"?", 0xFFEB009B, 214},  
02043 {"?", 0xFFBA0079, 215},  
02044 {"?", 0xFF82074E, 216},  
02045 {"?", 0xFFFFB8FF, 217},  
02046 {"?", 0xFFFFA63FF, 218},  
02047 {"?", 0xFFFFC1FFF, 219},  
02048 {"?", 0xFFD400B8, 220},  
02049 {"?", 0xFFB30098, 221},  
02050 {"?", 0xFF69005E, 222},  
02051 {"?", 0xFFFF8AFF, 223},  
02052 {"?", 0xFFFFC5EFF, 224},  
02053 {"?", 0xFFFFC2BFF, 225},  
02054 {"?", 0xFFFFF00FF, 226},  
02055 {"?", 0xFFCF00CO, 227},  
02056 {"?", 0xFF960090, 228},  
02057 {"?", 0xFF660057, 229},  
02058 {"?", 0xFFFFFA8FF, 230},  
02059 {"?", 0xFFFFC7AFF, 231},  
02060 {"?", 0xFFFF754FF, 232},  
02061 {"?", 0FFE300FF, 233},  
02062 {"?", 0xFFB100BD, 234},  
02063 {"?", 0xFF910099, 235},  
02064 {"?", 0xFFFFCB3FF, 236},  
02065 {"?", 0xFFFFABAFF, 2365},  
02066 {"?", 0xFFFF782FF, 237},  
02067 {"?", 0FFE66EFF, 2375},  
02068 {"?", 0xFFFF05EFF, 238},  
02069 {"?", 0xFFCF36FF, 2385},  
02070 {"?", 0FFE336FF, 239},  
02071 {"?", 0xFFBA0DFF, 2395},  
02072 {"?", 0xFFD10FFF, 240},  
02073 {"?", 0FFA800FF, 2405},  
02074 {"?", 0FFB600FA, 241},  
02075 {"?", 0FF9D00EB, 2415},  
02076 {"?", 0FFT50082, 242},  
02077 {"?", 0FF7700BD, 2425},  
02078 {"?", 0FFF2B5FF, 243},  
02079 {"?", 0FFE89EFF, 244},  
02080 {"?", 0FFDB78FF, 245},  
02081 {"?", 0FFB51AFF, 246},  
02082 {"?", 0FFA300FF, 247},  
02083 {"?", 0FF9600FA, 248},  
02084 {"?", 0FF6E00B8, 249},  
02085 {"?", 0FFF2D1FF, 250},  
02086 {"?", 0FFDE9CFF, 251},  
02087 {"?", 0FFC270FF, 252},  
02088 {"?", 0FF910DFF, 253},  
02089 {"?", 0FF8000FF, 254},  
02090 {"?", 0FF5E00BF, 255},  
02091 {"?", 0FFEDCCFF, 256},  
02092 {"?", 0FFCFA6FF, 2562},  
02093 {"?", 0FFC7ABFF, 2563},  
02094 {"?", 0FFB5A3FF, 2567},  
02095 {"?", 0FFDBA8FF, 257},  
02096 {"?", 0FFB387FF, 2572},  
02097 {"?", 0FFB391FF, 2573},  
02098 {"?", 0FF998CFF, 2577},  
02099 {"?", 0FF913DFF, 258},  
02100 {"?", 0FF8A47FF, 2582},  
02101 {"?", 0FF8A5EFF, 2583},  
02102 {"?", 0FF6957FF, 2587},  
02103 {"?", 0FF5F00D9, 259},  
02104 {"?", 0FF661AFF, 2592},  
02105 {"?", 0FF631CFF, 2593},  
02106 {"?", 0FF2600FF, 2597},  
02107 {"?", 0FF5B00BD, 260},  
02108 {"?", 0FF5C00F7, 2602},  
02109 {"?", 0FF4D00FA, 2603},  
02110 {"?", 0FF2D00ED, 2607},  
02111 {"?", 0FF500099, 261},  
02112 {"?", 0FF4F00DB, 2612},  
02113 {"?", 0FF5000D9, 2613},  
02114 {"?", 0FF2E00D9, 2617},  
02115 {"?", 0FF3F0073, 262},  
02116 {"?", 0FF3C008F, 2622},  
02117 {"?", 0FF4700AD, 2623},  
02118 {"?", 0FF2800B0, 2627},  
02119 {"?", 0FFE6DBFF, 263},  
02120 {"?", 0FFB8BAFF, 2635},
```

```
02121 {"?", 0xFFBDB8FF, 264},  
02122 {"?", 0xFF99A3FF, 2645},  
02123 {"?", 0xFF7570FF, 265},  
02124 {"?", 0xFF7582FF, 2655},  
02125 {"?", 0xFF361AFF, 266},  
02126 {"?", 0xFF6166FF, 2665},  
02127 {"?", 0xFF1C00FF, 267},  
02128 {"?", 0xFF2800E0, 268},  
02129 {"?", 0xFF0900E6, 2685},  
02130 {"?", 0xFF2600AB, 269},  
02131 {"?", 0xFF0C0082, 2695},  
02132 {"?", 0xFFB0BAFF, 270},  
02133 {"?", 0xFF99B3FF, 2705},  
02134 {"?", 0xFFCFE8FF, 2706},  
02135 {"?", 0xFFD4FOFF, 2707},  
02136 {"?", 0xFFBD6EFF, 2708},  
02137 {"?", 0xFF91A1FF, 271},  
02138 {"?", 0xFF6E8CFF, 2715},  
02139 {"?", 0xFF8CB5FF, 2716},  
02140 {"?", 0xFFB5E0FF, 2717},  
02141 {"?", 0xFF5496FF, 2718},  
02142 {"?", 0xFF6B85FF, 272},  
02143 {"?", 0xFF3B52FF, 2725},  
02144 {"?", 0xFF3657FF, 2726},  
02145 {"?", 0xFF4A94FF, 2727},  
02146 {"?", 0xFF0A4FFF, 2728},  
02147 {"?", 0xFF0009EB, 273},  
02148 {"?", 0xFF000DFF, 2735},  
02149 {"?", 0xFF0017FF, 2736},  
02150 {"?", 0xFF0020FA, 2738},  
02151 {"?", 0xFF0000B8, 274},  
02152 {"?", 0xFF000BD9, 2745},  
02153 {"?", 0xFF0012E6, 2746},  
02154 {"?", 0xFF001ED9, 2747},  
02155 {"?", 0xFF001AD9, 2748},  
02156 {"?", 0xFF030091, 275},  
02157 {"?", 0xFF0005B3, 2755},  
02158 {"?", 0xFF000BB5, 2756},  
02159 {"?", 0xFF0020B3, 2757},  
02160 {"?", 0xFF0026BD, 2758},  
02161 {"?", 0xFF020073, 276},  
02162 {"?", 0xFF00048C, 2765},  
02163 {"?", 0xFF000887, 2766},  
02164 {"?", 0xFF001A75, 2767},  
02165 {"?", 0xFF001F8F, 2768},  
02166 {"?", 0xFFBAEDFF, 277},  
02167 {"?", 0xFF9CDFFF, 278},  
02168 {"?", 0xFF52A8FF, 279},  
02169 {"?", 0xFF003BD1, 280},  
02170 {"?", 0xFF0031AD, 281},  
02171 {"?", 0xFF002675, 282},  
02172 {"?", 0xFFA6E8FF, 283},  
02173 {"?", 0xFF73CFFF, 284},  
02174 {"?", 0xFF1C91FF, 285},  
02175 {"?", 0xFF0055FA, 286},  
02176 {"?", 0xFF0048E0, 287},  
02177 {"?", 0xFF0041C4, 288},  
02178 {"?", 0xFF00246B, 289},  
02179 {"?", 0xFFBFFAFF, 290},  
02180 {"?", 0xFF96FAFF, 2905},  
02181 {"?", 0xFFABF7FF, 291},  
02182 {"?", 0xFF69EDFF, 2915},  
02183 {"?", 0xFF82E3FF, 292},  
02184 {"?", 0xFF26C2FF, 2925},  
02185 {"?", 0xFF006BFA, 293},  
02186 {"?", 0xFF008AFF, 2935},  
02187 {"?", 0xFF0055C9, 294},  
02188 {"?", 0xFF0079DB, 2945},  
02189 {"?", 0xFF0045A1, 295},  
02190 {"?", 0xFF0058A1, 2955},  
02191 {"?", 0xFF00294D, 296},  
02192 {"?", 0xFF00395C, 2965},  
02193 {"?", 0xFF82FCFF, 297},  
02194 {"?", 0xFFB3FFF2, 2975},  
02195 {"?", 0xFF4FEDFF, 298},  
02196 {"?", 0xFF69FFFF, 2985},  
02197 {"?", 0xFF26CFFF, 299},  
02198 {"?", 0xFF1AE3FF, 2995},  
02199 {"?", 0xFF008FFF, 300},  
02200 {"?", 0xFF00A0FA, 3005},  
02201 {"?", 0xFF0073D1, 301},  
02202 {"?", 0xFF0089CC, 3015},  
02203 {"?", 0xFF006080, 302},  
02204 {"?", 0xFF00687D, 3025},  
02205 {"?", 0xFF003B42, 303},  
02206 {"?", 0xFF004744, 3035},  
02207 {"?", 0xFFB3FFEB, 304},
```

```
02208 {"?", 0xFF7DFFE8, 305},  
02209 {"?", 0xFF40FFED, 306},  
02210 {"?", 0xFF009CBA, 307},  
02211 {"?", 0xFF008087, 308},  
02212 {"?", 0xFF004741, 309},  
02213 {"?", 0xFF91FFE6, 310},  
02214 {"?", 0xFF91FFE0, 3105},  
02215 {"?", 0xFF5EFE0, 311},  
02216 {"?", 0xFF5EFD1, 3115},  
02217 {"?", 0xFF0AFCE3, 312},  
02218 {"?", 0xFF2BFFC9, 3125},  
02219 {"?", 0xFF00DECC, 313},  
02220 {"?", 0xFF00E8C3, 3135},  
02221 {"?", 0xFF00B3A2, 314},  
02222 {"?", 0xFF00C49F, 3145},  
02223 {"?", 0xFF009180, 315},  
02224 {"?", 0xFF009E78, 3155},  
02225 {"?", 0xFF00523C, 316},  
02226 {"?", 0xFF005940, 3165},  
02227 {"?", 0xFFD1FFEB, 317},  
02228 {"?", 0xFF9EFFD9, 318},  
02229 {"?", 0xFF7AFFCF, 319},  
02230 {"?", 0xFF00EDA4, 320},  
02231 {"?", 0xFF00C487, 321},  
02232 {"?", 0xFF00A66F, 322},  
02233 {"?", 0xFF008754, 323},  
02234 {"?", 0xFFB8FFE0, 324},  
02235 {"?", 0xFFA1FFD1, 3242},  
02236 {"?", 0xFFA8FFCF, 3245},  
02237 {"?", 0xFF91FFC2, 3248},  
02238 {"?", 0xFF70FFBD, 325},  
02239 {"?", 0xFF87FFC2, 3252},  
02240 {"?", 0xFF82FFB8, 3255},  
02241 {"?", 0xFF69FFAB, 3258},  
02242 {"?", 0xFF21FF9E, 326},  
02243 {"?", 0xFF4AFFAB, 3262},  
02244 {"?", 0xFF4FFFA1, 3265},  
02245 {"?", 0xFF1AF82, 3268},  
02246 {"?", 0xFF00D477, 327},  
02247 {"?", 0xFF00FF8F, 3272},  
02248 {"?", 0xFF0DF87, 3275},  
02249 {"?", 0xFF00F26D, 3278},  
02250 {"?", 0xFF00AD5F, 328},  
02251 {"?", 0xFF00D975, 3282},  
02252 {"?", 0xFF00ED77, 3285},  
02253 {"?", 0xFF00CC5E, 3288},  
02254 {"?", 0xFF008A4A, 329},  
02255 {"?", 0xFF008A46, 3292},  
02256 {"?", 0xFF00C95F, 3295},  
02257 {"?", 0xFF009440, 3298},  
02258 {"?", 0xFF006635, 330},  
02259 {"?", 0xFF004F24, 3302},  
02260 {"?", 0xFF006327, 3305},  
02261 {"?", 0xFF00471D, 3308},  
02262 {"?", 0xFFC2FFD6, 331},  
02263 {"?", 0xFFB3FFCC, 332},  
02264 {"?", 0xFF91FFBA, 333},  
02265 {"?", 0xFF00F763, 334},  
02266 {"?", 0xFF00B33E, 335},  
02267 {"?", 0xFF00872D, 336},  
02268 {"?", 0xFFB0FFCC, 337},  
02269 {"?", 0xFFA6FFBF, 3375},  
02270 {"?", 0xFF87FFAD, 338},  
02271 {"?", 0xFF8CFFAB, 3385},  
02272 {"?", 0xFF29FF70, 339},  
02273 {"?", 0xFF63FF8C, 3395},  
02274 {"?", 0xFF00E84F, 340},  
02275 {"?", 0xFF26FF59, 3405},  
02276 {"?", 0xFF00B53C, 341},  
02277 {"?", 0xFF00C72E, 3415},  
02278 {"?", 0xFF00912A, 342},  
02279 {"?", 0xFF009421, 3425},  
02280 {"?", 0xFF02631C, 343},  
02281 {"?", 0xFF005710, 3435},  
02282 {"?", 0xFFBAFFC4, 344},  
02283 {"?", 0xFF9EFFAD, 345},  
02284 {"?", 0xFF73FF87, 346},  
02285 {"?", 0xFF00F723, 347},  
02286 {"?", 0xFF00C21D, 348},  
02287 {"?", 0xFF00940D, 349},  
02288 {"?", 0xFF0D4000, 350},  
02289 {"?", 0xFFD4FFD6, 351},  
02290 {"?", 0xFFBAFFBF, 352},  
02291 {"?", 0xFF9EFFA3, 353},  
02292 {"?", 0xFF33FF1A, 354},  
02293 {"?", 0xFF0FF00, 355},  
02294 {"?", 0xFF09BA00, 356},
```

```
02295 {"?", 0xFF167000, 357},  
02296 {"?", 0xFFBAFF9E, 358},  
02297 {"?", 0xFFA3FF82, 359},  
02298 {"?", 0xFF6BF33, 360},  
02299 {"?", 0xFF4FFF00, 361},  
02300 {"?", 0xFF46E800, 362},  
02301 {"?", 0xFF3EC200, 363},  
02302 {"?", 0xFF349400, 364},  
02303 {"?", 0xFFEOFBB5, 365},  
02304 {"?", 0xFFCCFF8F, 366},  
02305 {"?", 0xFFADFF69, 367},  
02306 {"?", 0xFF6EF00, 368},  
02307 {"?", 0xFF61ED00, 369},  
02308 {"?", 0xFF52BA00, 370},  
02309 {"?", 0xFF407000, 371},  
02310 {"?", 0xFFE6FFAB, 372},  
02311 {"?", 0xFFD6FF8A, 373},  
02312 {"?", 0xFFC2FF6E, 374},  
02313 {"?", 0xFF96FF38, 375},  
02314 {"?", 0xFF74F200, 376},  
02315 {"?", 0xFF6BC200, 377},  
02316 {"?", 0xFF436600, 378},  
02317 {"?", 0xFFE8FF6B, 379},  
02318 {"?", 0xFFDEF47, 380},  
02319 {"?", 0xFFCCFF17, 381},  
02320 {"?", 0xFFB5FF00, 382},  
02321 {"?", 0xFFA5CF00, 383},  
02322 {"?", 0xFF90B000, 384},  
02323 {"?", 0xFF686B00, 385},  
02324 {"?", 0xFFF0FF70, 386},  
02325 {"?", 0xFFE6FF42, 387},  
02326 {"?", 0xFFDBFF36, 388},  
02327 {"?", 0xFFCCFF26, 389},  
02328 {"?", 0xFFB7EB00, 390},  
02329 {"?", 0xFF95AB00, 391},  
02330 {"?", 0xFF798200, 392},  
02331 {"?", 0xFFFF7FF73, 393},  
02332 {"?", 0xFFFFCF52, 3935},  
02333 {"?", 0xFFFF0FF3D, 394},  
02334 {"?", 0xFFF7FF26, 3945},  
02335 {"?", 0xFFEBFF26, 395},  
02336 {"?", 0xFFF0FF00, 3955},  
02337 {"?", 0xFFE3FF0F, 396},  
02338 {"?", 0xFFEBFF00, 3965},  
02339 {"?", 0xFFCE300, 397},  
02340 {"?", 0xFFB5B500, 3975},  
02341 {"?", 0xFFABB800, 398},  
02342 {"?", 0xFF969200, 3985},  
02343 {"?", 0xFF919100, 399},  
02344 {"?", 0xFF5C5900, 3995},  
02345 {"?", 0xFFD6D0C9, 400},  
02346 {"?", 0xFFC4BBAF, 401},  
02347 {"?", 0xFFB0A597, 402},  
02348 {"?", 0xFF918779, 403},  
02349 {"?", 0xFF706758, 404},  
02350 {"?", 0xFF474030, 405},  
02351 {"?", 0xFFD6CBC9, 406},  
02352 {"?", 0xFFBDAEAC, 407},  
02353 {"?", 0xFFA89796, 408},  
02354 {"?", 0xFF8C7A77, 409},  
02355 {"?", 0xFF705C59, 410},  
02356 {"?", 0xFF47342B, 411},  
02357 {"?", 0xFF050402, 412},  
02358 {"?", 0xFFCCCCBA, 413},  
02359 {"?", 0xFFB3B3A1, 414},  
02360 {"?", 0xFF969684, 415},  
02361 {"?", 0xFF80806B, 416},  
02362 {"?", 0xFF585943, 417},  
02363 {"?", 0xFF3E402C, 418},  
02364 {"?", 0xFF000000, 419},  
02365 {"?", 0xFFD9D9D9, 420},  
02366 {"?", 0xFFBDBDBD, 421},  
02367 {"?", 0xFFABABAB, 422},  
02368 {"?", 0xFF8F8F8F, 423},  
02369 {"?", 0xFF636363, 424},  
02370 {"?", 0xFFB3B3B3, 425},  
02371 {"?", 0xFF000000, 426}, /* TODO: duplicate case value */  
02372 {"?", 0xFFE3E3E3, 427},  
02373 {"?", 0xFFCDD1D1, 428},  
02374 {"?", 0xFFA8ADAD, 429},  
02375 {"?", 0xFF858C8C, 430},  
02376 {"?", 0xFF525B5C, 431},  
02377 {"?", 0xFF2D393B, 432},  
02378 {"?", 0xFF090C0D, 433},  
02379 {"?", 0xFFEDE6E8, 434},  
02380 {"?", 0xFFDED6DB, 435},  
02381 {"?", 0xFFC2BFBF, 436},
```

```
02382 {"?", 0xFF8A8C8A, 437},  
02383 {"?", 0xFF394500, 438},  
02384 {"?", 0xFF293300, 439},  
02385 {"?", 0xFF202700, 440},  
02386 {"?", 0xFFDAE8D8, 441},  
02387 {"?", 0xFFBECFB8, 442},  
02388 {"?", 0xFF9DB39D, 443},  
02389 {"?", 0xFF7E947E, 444},  
02390 {"?", 0xFF475947, 445},  
02391 {"?", 0xFF324031, 446},  
02392 {"?", 0xFF272E20, 447},  
02393 {"?", 0xFF2D3E00, 448},  
02394 {"?", 0xFF4F3A00, 4485},  
02395 {"?", 0xFF3D5200, 449},  
02396 {"?", 0xFF8A6E07, 4495},  
02397 {"?", 0xFF506700, 450},  
02398 {"?", 0xFFA38B24, 4505},  
02399 {"?", 0xFFAABB573, 451},  
02400 {"?", 0xFFC2B061, 4515},  
02401 {"?", 0xFFC2CF9C, 452},  
02402 {"?", 0xFFD4C581, 4525},  
02403 {"?", 0xFFDBE3BF, 453},  
02404 {"?", 0xFFE3DA9F, 4535},  
02405 {"?", 0xFFE8EDD6, 454},  
02406 {"?", 0xFFF0E9C2, 4545},  
02407 {"?", 0xFF594A00, 455},  
02408 {"?", 0xFF917C00, 456},  
02409 {"?", 0xFFB89C00, 457},  
02410 {"?", 0xFFE6E645, 458},  
02411 {"?", 0xFFF0ED73, 459},  
02412 {"?", 0xFFFF5F28F, 460},  
02413 {"?", 0xFFF7F7A6, 461},  
02414 {"?", 0xFF402600, 462},  
02415 {"?", 0xFF361500, 4625},  
02416 {"?", 0xFF6B3D00, 463},  
02417 {"?", 0xFF8F4A06, 4635},  
02418 {"?", 0xFF955300, 464},  
02419 {"?", 0xFFB8743B, 4645},  
02420 {"?", 0xFFCCAD6B, 465},  
02421 {"?", 0xFFD19B73, 4655},  
02422 {"?", 0xFFE0C791, 466},  
02423 {"?", 0FFE6BC9C, 4665},  
02424 {"?", 0FFE8D9A8, 467},  
02425 {"?", 0FFF0D5BD, 4675},  
02426 {"?", 0FFF0E8C4, 468},  
02427 {"?", 0FFF5E4D3, 4685},  
02428 {"?", 0FF4A1A00, 469},  
02429 {"?", 0FF420D00, 4695},  
02430 {"?", 0FFAB4800, 470},  
02431 {"?", 0FF823126, 4705},  
02432 {"?", 0FFD15600, 471},  
02433 {"?", 0FFA8625D, 4715},  
02434 {"?", 0FFFFFA87A, 472},  
02435 {"?", 0FFBF827C, 4725},  
02436 {"?", 0FFFFC4A3, 473},  
02437 {"?", 0FFD9A9A7, 4735},  
02438 {"?", 0FFFFD9BD, 474},  
02439 {"?", 0FFE6BEBC, 4745},  
02440 {"?", 0FFFFE3CC, 475},  
02441 {"?", 0FFF0D8D3, 4755},  
02442 {"?", 0FF381C00, 476},  
02443 {"?", 0FF4F1800, 477},  
02444 {"?", 0FF6B1200, 478},  
02445 {"?", 0FFB08573, 479},  
02446 {"?", 0FFD9B5B0, 480},  
02447 {"?", 0FFE8CF89, 481},  
02448 {"?", 0FFF2E0DE, 482},  
02449 {"?", 0FF660700, 483},  
02450 {"?", 0FFB50900, 484},  
02451 {"?", 0FFF0D00, 485},  
02452 {"?", 0FFF8796, 486},  
02453 {"?", 0FFFFA6B8, 487},  
02454 {"?", 0FFFFBDCF, 488},  
02455 {"?", 0FFFFD9E3, 489},  
02456 {"?", 0FF471300, 490},  
02457 {"?", 0FF7A1A00, 491},  
02458 {"?", 0FF942200, 492},  
02459 {"?", 0FFF283BB, 493},  
02460 {"?", 0FFFABDE, 494},  
02461 {"?", 0FFFC2E3, 495},  
02462 {"?", 0FFFD6E8, 496},  
02463 {"?", 0FF381100, 497},  
02464 {"?", 0FF330E00, 4975},  
02465 {"?", 0FF662500, 498},  
02466 {"?", 0FF853241, 4985},  
02467 {"?", 0FF853100, 499},  
02468 {"?", 0FFA85868, 4995},
```

```
02469 {"?", 0xFFE38DB3, 500},  
02470 {"?", 0xFFC47A8F, 5005},  
02471 {"?", 0xFFF7B5D7, 501},  
02472 {"?", 0FFE3AAC1, 5015},  
02473 {"?", 0xFFFFCCFE3, 502},  
02474 {"?", 0xFFEDC2D1, 5025},  
02475 {"?", 0xFFFFE3EB, 503},  
02476 {"?", 0xFFF7DFE1, 5035},  
02477 {"?", 0xFF320000, 504},  
02478 {"?", 0xFF600000, 505},  
02479 {"?", 0xFF770000, 506},  
02480 {"?", 0xFFDE82C4, 507},  
02481 {"?", 0xFFF2A3E3, 508},  
02482 {"?", 0xFFFFC2ED, 509},  
02483 {"?", 0xFFFFD4F0, 510},  
02484 {"?", 0FFD0066, 511},  
02485 {"?", 0xFF2B0041, 5115},  
02486 {"?", 0FF6100CE, 512},  
02487 {"?", 0FF592482, 5125},  
02488 {"?", 0FF8A1FFF, 513},  
02489 {"?", 0FF8257B8, 5135},  
02490 {"?", 0FFD980FF, 514},  
02491 {"?", 0FFB38CE0, 5145},  
02492 {"?", 0FFED9EFF, 515},  
02493 {"?", 0FFD4B3EB, 5155},  
02494 {"?", 0FFF7BAFF, 516},  
02495 {"?", 0FFE8CFF2, 5165},  
02496 {"?", 0FFFFD1FF, 517},  
02497 {"?", 0FFF2E0F7, 5175},  
02498 {"?", 0FF2E005C, 518},  
02499 {"?", 0FFC0022, 5185},  
02500 {"?", 0FF44009D, 519},  
02501 {"?", 0FF3D0C4E, 5195},  
02502 {"?", 0FF5C00E0, 520},  
02503 {"?", 0FF7A5E85, 5205},  
02504 {"?", 0FFBA87FF, 521},  
02505 {"?", 0FB59EC2, 5215},  
02506 {"?", 0FFD4A1FF, 522},  
02507 {"?", 0FFD4BAD9, 5225},  
02508 {"?", 0FFE6BDFF, 523},  
02509 {"?", 0FFE6D4E6, 5235},  
02510 {"?", 0FFF0D9FF, 524},  
02511 {"?", 0FFF0E6ED, 5245},  
02512 {"?", 0FF270085, 525},  
02513 {"?", 0FFD0B4D, 5255},  
02514 {"?", 0FF3B00ED, 526},  
02515 {"?", 0FF20258A, 5265},  
02516 {"?", 0FF4500FF, 527},  
02517 {"?", 0FF3848A8, 5275},  
02518 {"?", 0FF9673FF, 528},  
02519 {"?", 0FF7280C4, 5285},  
02520 {"?", 0FFBD99FF, 529},  
02521 {"?", 0FFA8B3E6, 5295},  
02522 {"?", 0FFD1B0FF, 530},  
02523 {"?", 0FFC7CEED, 5305},  
02524 {"?", 0FFE6CCFF, 531},  
02525 {"?", 0FFE4E4F2, 5315},  
02526 {"?", 0FF00193F, 532},  
02527 {"?", 0FF00227B, 533},  
02528 {"?", 0FF002CAA, 534},  
02529 {"?", 0FF94B3ED, 535},  
02530 {"?", 0FFB0C7F2, 536},  
02531 {"?", 0FFC7DBF7, 537},  
02532 {"?", 0FFDEE8FA, 538},  
02533 {"?", 0FF00274D, 539},  
02534 {"?", 0FF00223D, 5395},  
02535 {"?", 0FF003473, 540},  
02536 {"?", 0FF3A728A, 5405},  
02537 {"?", 0FF00449E, 541},  
02538 {"?", 0FF5A8A96, 5415},  
02539 {"?", 0FF5EC1F7, 542},  
02540 {"?", 0FF79A6AD, 5425},  
02541 {"?", 0FF96E3FF, 543},  
02542 {"?", 0FFB8CDD4, 5435},  
02543 {"?", 0FFB3FOFF, 544},  
02544 {"?", 0FFCCDCDE, 5445},  
02545 {"?", 0FFC7F7FF, 545},  
02546 {"?", 0FFDAE8E8, 5455},  
02547 {"?", 0FF02272B, 546},  
02548 {"?", 0FF002B24, 5463},  
02549 {"?", 0FF000D09, 5467},  
02550 {"?", 0FF003440, 547},  
02551 {"?", 0FF167A58, 5473},  
02552 {"?", 0FF1D4230, 5477},  
02553 {"?", 0FF00465C, 548},  
02554 {"?", 0FF43B08B, 5483},  
02555 {"?", 0FF48705D, 5487},
```

```
02556 {"?", 0xFF56ADBA, 549},  
02557 {"?", 0xFF73C9AD, 5493},  
02558 {"?", 0xFF829E90, 5497},  
02559 {"?", 0xFF7BC1C9, 550},  
02560 {"?", 0xFF9CDCB5, 5503},  
02561 {"?", 0xFFA1B5A8, 5507},  
02562 {"?", 0xFFA2D7DE, 551},  
02563 {"?", 0xFFC7F2E1, 5513},  
02564 {"?", 0xFFBED1C5, 5517},  
02565 {"?", 0xFFC5E8E8, 552},  
02566 {"?", 0xFFDCFC7EB, 5523},  
02567 {"?", 0xFFD5E3DA, 5527},  
02568 {"?", 0xFF143319, 553},  
02569 {"?", 0xFF102E14, 5535},  
02570 {"?", 0xFF115422, 554},  
02571 {"?", 0xFF327A3D, 5545},  
02572 {"?", 0xFF187031, 555},  
02573 {"?", 0xFF5A9E68, 5555},  
02574 {"?", 0xFF66BA80, 556},  
02575 {"?", 0xFF84BD8F, 5565},  
02576 {"?", 0xFF98D9AD, 557},  
02577 {"?", 0xFFA9D4B2, 5575},  
02578 {"?", 0xFFBAE8CA, 558},  
02579 {"?", 0xFFCAE6CC, 5585},  
02580 {"?", 0xFFCEF0D8, 559},  
02581 {"?", 0xFFDDEDDA, 5595},  
02582 {"?", 0xFF0D4018, 560},  
02583 {"?", 0xFF050F07, 5605},  
02584 {"?", 0xFF127A38, 561},  
02585 {"?", 0xFF2E522B, 5615},  
02586 {"?", 0xFF1AB058, 562},  
02587 {"?", 0xFF5A7D57, 5625},  
02588 {"?", 0xFF79FCAC, 563},  
02589 {"?", 0xFF89A386, 5635},  
02590 {"?", 0xFFA1FFCC, 564},  
02591 {"?", 0xFFAEBFA6, 5645},  
02592 {"?", 0xFFC4FFDE, 565},  
02593 {"?", 0xFFC5D1BE, 5655},  
02594 {"?", 0xFFDBFFE8, 566},  
02595 {"?", 0xFFDAE6D5, 5665},  
02596 {"?", 0xFF0E4D1C, 567},  
02597 {"?", 0xFF14A346, 568},  
02598 {"?", 0xFF04D45B, 569},  
02599 {"?", 0xFF85FFB5, 570},  
02600 {"?", 0xFFADFFCF, 571},  
02601 {"?", 0xFFC4FFDB, 572},  
02602 {"?", 0xFFDBFFE8, 573}, /* TODO: duplicate case value */  
02603 {"?", 0xFF314A0E, 574},  
02604 {"?", 0xFF1F2E07, 5743},  
02605 {"?", 0xFF243600, 5747},  
02606 {"?", 0xFF3E7800, 575},  
02607 {"?", 0xFF3F5410, 5753},  
02608 {"?", 0xFF547306, 5757},  
02609 {"?", 0xFF4F9C00, 576},  
02610 {"?", 0xFF5C6E1D, 5763},  
02611 {"?", 0xFF849C32, 5767},  
02612 {"?", 0xFFAEE67C, 577},  
02613 {"?", 0xFF909E5A, 5773},  
02614 {"?", 0xFFA5B85E, 5777},  
02615 {"?", 0xFFC0F090, 578},  
02616 {"?", 0xFFAFBA86, 5783},  
02617 {"?", 0xFFCEDE99, 5787},  
02618 {"?", 0xFFCDF7A3, 579},  
02619 {"?", 0xFFC9D1A5, 5793},  
02620 {"?", 0xFFDCE8B0, 5797},  
02621 {"?", 0xFFDCFAB9, 580},  
02622 {"?", 0xFFDEE3C8, 5803},  
02623 {"?", 0FFE9F0CE, 5807},  
02624 {"?", 0xFF464700, 581},  
02625 {"?", 0xFF363605, 5815},  
02626 {"?", 0xFF788A00, 582},  
02627 {"?", 0xFF69660E, 5825},  
02628 {"?", 0xFFA3D400, 583},  
02629 {"?", 0xFF999632, 5835},  
02630 {"?", 0xFFD3F032, 584},  
02631 {"?", 0xFFB3B15F, 5845},  
02632 {"?", 0xFFDEFA55, 585},  
02633 {"?", 0xFFD1D190, 5855},  
02634 {"?", 0FFE8FF78, 586},  
02635 {"?", 0FFDEDEA6, 5865},  
02636 {"?", 0FFF2FF99, 587},  
02637 {"?", 0FFE8EBC0, 5875},  
02638 {"?", 0xFFFFFB5, 600},  
02639 {"?", 0xFFFFFFF99, 601},  
02640 {"?", 0xFFFFFFF7D, 602}, /* TODO: duplicate case value */  
02641 {"?", 0xFFFFFC4E, 603},  
02642 {"?", 0FFF7F71E, 604},
```

```
02643 {"?", 0xFFEDE800, 605},  
02644 {"?", 0xFFE0D700, 606},  
02645 {"?", 0xFFFFCFCFF, 607},  
02646 {"?", 0xFFFFAFAAA, 608},  
02647 {"?", 0xFFF5F584, 609},  
02648 {"?", 0xFFFF0F065, 610},  
02649 {"?", 0xFFE3E112, 611},  
02650 {"?", 0xFFCCC800, 612},  
02651 {"?", 0xFFB3AB00, 613},  
02652 {"?", 0xFFFF5F5C4, 614},  
02653 {"?", 0xFFFF0EDAF, 615},  
02654 {"?", 0xFFE8E397, 616},  
02655 {"?", 0xFFD4CF6E, 617},  
02656 {"?", 0xFFB3AD17, 618},  
02657 {"?", 0xFF918C00, 619},  
02658 {"?", 0xFF787200, 620},  
02659 {"?", 0xFFD9FAE1, 621},  
02660 {"?", 0xFFBAF5C6, 622},  
02661 {"?", 0xFF9CE6AE, 623},  
02662 {"?", 0xFF72CC85, 624},  
02663 {"?", 0xFF4BAB60, 625},  
02664 {"?", 0xFF175E22, 626},  
02665 {"?", 0xFF04290A, 627},  
02666 {"?", 0xFFCFFFF0, 628},  
02667 {"?", 0xFFA8FFE8, 629},  
02668 {"?", 0xFF87FFE3, 630},  
02669 {"?", 0xFF52FADC, 631},  
02670 {"?", 0xFF13F2CE, 632},  
02671 {"?", 0xFF00BFAC, 633},  
02672 {"?", 0xFF00998B, 634},  
02673 {"?", 0xFFADFFEB, 635},  
02674 {"?", 0xFF8CFEE8, 636},  
02675 {"?", 0xFF73FFE8, 637},  
02676 {"?", 0xFF2BFEE6, 638},  
02677 {"?", 0xFF00F2E6, 639},  
02678 {"?", 0xFF00C7C7, 640},  
02679 {"?", 0xFF00ABB3, 641},  
02680 {"?", 0xFFD2F0FA, 642},  
02681 {"?", 0xFFB8E4F5, 643},  
02682 {"?", 0xFF8BCCF0, 644},  
02683 {"?", 0xFF64A7E8, 645},  
02684 {"?", 0xFF4696E3, 646},  
02685 {"?", 0xFF0056C4, 647},  
02686 {"?", 0xFF002D75, 648},  
02687 {"?", 0xFFD9EDFC, 649},  
02688 {"?", 0xFFBEE3FA, 650},  
02689 {"?", 0xFF95C5F0, 651},  
02690 {"?", 0xFF5C97E6, 652},  
02691 {"?", 0xFF004ECC, 653},  
02692 {"?", 0xFF00399E, 654},  
02693 {"?", 0xFF002B7A, 655},  
02694 {"?", 0xFFDBF5FF, 656},  
02695 {"?", 0xFFC2EBFF, 657},  
02696 {"?", 0xFF96CCFF, 658},  
02697 {"?", 0xFF5CA6FF, 659},  
02698 {"?", 0xFF1A6EFF, 660},  
02699 {"?", 0xFF0048E8, 661},  
02700 {"?", 0xFF003BD1, 662}, /* TODO: duplicate case value */  
02701 {"?", 0xFFEDFOFF, 663},  
02702 {"?", 0xFFE3E8FF, 664},  
02703 {"?", 0xFFC8CFFA, 665},  
02704 {"?", 0xFFA4A6ED, 666},  
02705 {"?", 0xFF6970DB, 667},  
02706 {"?", 0xFF3E40B3, 668},  
02707 {"?", 0xFF201E87, 669},  
02708 {"?", 0xFFFFDDEFF, 670},  
02709 {"?", 0xFFFFCCFF, 671},  
02710 {"?", 0xFFFF7A8FF, 672},  
02711 {"?", 0xFFF082FF, 673},  
02712 {"?", 0FFE854FF, 674},  
02713 {"?", 0xFFCD00F7, 675},  
02714 {"?", 0xFFBB00C7, 676},  
02715 {"?", 0xFFFFADEFF, 677},  
02716 {"?", 0xFFFF7C9FF, 678},  
02717 {"?", 0xFFF2BAFF, 679},  
02718 {"?", 0FFE18EFA, 680},  
02719 {"?", 0xFFC15FF5, 681},  
02720 {"?", 0xFFA82FE0, 682},  
02721 {"?", 0xFF810091, 683},  
02722 {"?", 0xFFFFACFFA, 684},  
02723 {"?", 0xFFFF7BAF7, 685},  
02724 {"?", 0xFFFF2AAF2, 686},  
02725 {"?", 0xFFDC7EE0, 687},  
02726 {"?", 0xFFC459CF, 688},  
02727 {"?", 0xFF9D27A8, 689},  
02728 {"?", 0xFF690369, 690},  
02729 {"?", 0xFFFFCD7E8, 691},
```

```
02730 {"?", 0xFFFFAC0E1, 692},  
02731 {"?", 0xFFFF0A8D3, 693},  
02732 {"?", 0xFFE683BA, 694},  
02733 {"?", 0xFFFFBF508A, 695},  
02734 {"?", 0xFF991846, 696},  
02735 {"?", 0xFF7D0925, 697},  
02736 {"?", 0xFFFFD6EB, 698},  
02737 {"?", 0xFFFFFC2E6, 699},  
02738 {"?", 0xFFFFA3DB, 700},  
02739 {"?", 0xFFFF78CC, 701},  
02740 {"?", 0xFFF24BA0, 702},  
02741 {"?", 0xFFD62463, 703},  
02742 {"?", 0xFFBA0025, 704},  
02743 {"?", 0xFFFFE8F2, 705},  
02744 {"?", 0xFFFFD4E6, 706},  
02745 {"?", 0xFFFFB3DB, 707},  
02746 {"?", 0xFFFF8ACT, 708},  
02747 {"?", 0xFFFF579E, 709},  
02748 {"?", 0xFFFF366B, 710},  
02749 {"?", 0xFFFFA0032, 711},  
02750 {"?", 0xFFFFDBB0, 712},  
02751 {"?", 0xFFFFCF96, 713},  
02752 {"?", 0xFFFFFB875, 714},  
02753 {"?", 0xFFFFA14A, 715},  
02754 {"?", 0xFFFF8717, 716},  
02755 {"?", 0xFFFFA7000, 717},  
02756 {"?", 0xFFEB6300, 718},  
02757 {"?", 0xFFFFE6BF, 719},  
02758 {"?", 0xFFFFCD7A7, 720},  
02759 {"?", 0xFFF7BC77, 721},  
02760 {"?", 0FFE89538, 722},  
02761 {"?", 0FFD4740B, 723},  
02762 {"?", 0FFA14C00, 724},  
02763 {"?", 0FF823B00, 725},  
02764 {"?", 0FFFAE6C0, 726},  
02765 {"?", 0FFF2CEA0, 727},  
02766 {"?", 0FFE6B577, 728},  
02767 {"?", 0FFD19052, 729},  
02768 {"?", 0FFB56E2B, 730},  
02769 {"?", 0FF753700, 731},  
02770 {"?", 0FF5C2800, 732},  
02771 {"?", 0FFFFF5D1, 7401},  
02772 {"?", 0FFFFF0B3, 7402},  
02773 {"?", 0FFFE680, 7403},  
02774 {"?", 0FFFFE833, 7404}, /* TODO: duplicate case value */  
02775 {"?", 0FFFFE600, 7405}, /* TODO: duplicate case value */  
02776 {"?", 0xFFFFD100, 7406},  
02777 {"?", 0FFE3B122, 7407},  
02778 {"?", 0xFFFFBF0D, 7408},  
02779 {"?", 0xFFFFB30D, 7409},  
02780 {"?", 0xFFFFB373, 7410},  
02781 {"?", 0FFFA64F, 7411},  
02782 {"?", 0FFED8A00, 7412},  
02783 {"?", 0FF57300, 7413},  
02784 {"?", 0FFE37B00, 7414},  
02785 {"?", 0FFFD1D9, 7415},  
02786 {"?", 0FFF6666, 7416},  
02787 {"?", 0FFF4040, 7417},  
02788 {"?", 0FF24961, 7418},  
02789 {"?", 0FFD15473, 7419},  
02790 {"?", 0FFC22976, 7420},  
02791 {"?", 0FF630046, 7421},  
02792 {"?", 0FFFE8F2, 7422}, /* TODO: duplicate case value */  
02793 {"?", 0FFF73C7, 7423}, /* TODO: duplicate case value */  
02794 {"?", 0FF40B3, 7424},  
02795 {"?", 0FFED18A6, 7425},  
02796 {"?", 0FFD10073, 7426},  
02797 {"?", 0FFB80040, 7427},  
02798 {"?", 0FF73173F, 7428},  
02799 {"?", 0FFFD1F7, 7429},  
02800 {"?", 0FFFAB0FF, 7430},  
02801 {"?", 0FFF296ED, 7431},  
02802 {"?", 0FFE667DF, 7432},  
02803 {"?", 0FFD936B8, 7433},  
02804 {"?", 0FFCC29AD, 7434},  
02805 {"?", 0FFA60095, 7435},  
02806 {"?", 0FFF7EBFF, 7436},  
02807 {"?", 0FFF0CCFF, 7437},  
02808 {"?", 0FFD9A6FF, 7438},  
02809 {"?", 0FFCCA6FF, 7439},  
02810 {"?", 0FFB399FF, 7440},  
02811 {"?", 0FFA380FF, 7441},  
02812 {"?", 0FF804DFF, 7442},  
02813 {"?", 0FFF0F2FF, 7443},  
02814 {"?", 0FFCCD4FF, 7444},  
02815 {"?", 0FFADC6F7, 7445},  
02816 {"?", 0FF919EFF, 7446},
```

```
02817 {"?", 0xFF5357CF, 7447},  
02818 {"?", 0xFF4E4373, 7448},  
02819 {"?", 0xFF270020, 7449},  
02820 {"?", 0xFFCCE6FF, 7450},  
02821 {"?", 0xFF99C9FF, 7451},  
02822 {"?", 0xFF80ADFF, 7452},  
02823 {"?", 0xFF80BDFF, 7453},  
02824 {"?", 0xFF73AEE6, 7454},  
02825 {"?", 0xFF3378FF, 7455},  
02826 {"?", 0xFF6B9AED, 7456},  
02827 {"?", 0xFFE0FFFA, 7457},  
02828 {"?", 0xFF90F0E4, 7458},  
02829 {"?", 0xFF5FDDED1, 7459},  
02830 {"?", 0xFF00F2F2, 7460},  
02831 {"?", 0xFF38B8FF, 7461},  
02832 {"?", 0xFF0073E6, 7462},  
02833 {"?", 0xFF003359, 7463},  
02834 {"?", 0xFFFFFFFE6, 7464},  
02835 {"?", 0xFF80FFBF, 7465},  
02836 {"?", 0xFF4DFFC4, 7466},  
02837 {"?", 0xFF0DFFBF, 7467},  
02838 {"?", 0xFF00A5B8, 7468},  
02839 {"?", 0xFF007A99, 7469},  
02840 {"?", 0xFF1C778C, 7470},  
02841 {"?", 0xFFB8FFDB, 7471},  
02842 {"?", 0xFF7AFFBF, 7472},  
02843 {"?", 0xFF46EB91, 7473},  
02844 {"?", 0xFF14C78F, 7474},  
02845 {"?", 0xFF59B386, 7475},  
02846 {"?", 0xFF00663A, 7476},  
02847 {"?", 0xFFL05249, 7477},  
02848 {"?", 0xFFD1FFDB, 7478},  
02849 {"?", 0xFF73FF80, 7479},  
02850 {"?", 0xFF66FF80, 7480},  
02851 {"?", 0xFF66FF73, 7481},  
02852 {"?", 0xFF33FF40, 7482},  
02853 {"?", 0xFF117300, 7483},  
02854 {"?", 0xFF008013, 7484},  
02855 {"?", 0xFFFF0FFE6, 7485},  
02856 {"?", 0xFFCCFB3, 7486},  
02857 {"?", 0xFFB3FF8C, 7487},  
02858 {"?", 0xFF91FF66, 7488},  
02859 {"?", 0xFFFF5ED2F, 7489},  
02860 {"?", 0xFF5BA621, 7490},  
02861 {"?", 0xFF689900, 7491},  
02862 {"?", 0xFFD1ED77, 7492},  
02863 {"?", 0xFFC5E693, 7493},  
02864 {"?", 0xFFA3D982, 7494},  
02865 {"?", 0xFF86B324, 7495},  
02866 {"?", 0xFF5F9E00, 7496},  
02867 {"?", 0xFF738639, 7497},  
02868 {"?", 0xFF263300, 7498},  
02869 {"?", 0xFFFFFDAD9, 7499},  
02870 {"?", 0xFFF7F2D2, 7500},  
02871 {"?", 0xFFFF0E6C0, 7501},  
02872 {"?", 0xFFE6D395, 7502},  
02873 {"?", 0xFFBFA87C, 7503},  
02874 {"?", 0xFF997354, 7504},  
02875 {"?", 0xFF735022, 7505},  
02876 {"?", 0xFFFFF2D9, 7506},  
02877 {"?", 0xFFFFE6B3, 7507},  
02878 {"?", 0xFFF5D093, 7508},  
02879 {"?", 0xFFF2C279, 7509},  
02880 {"?", 0xFFE39F40, 7510},  
02881 {"?", 0xFFBF6900, 7511},  
02882 {"?", 0xFFAB5C00, 7512},  
02883 {"?", 0xFFF7CBB2, 7513},  
02884 {"?", 0xFFF2B896, 7514},  
02885 {"?", 0FFE09270, 7515},  
02886 {"?", 0FFA65000, 7516},  
02887 {"?", 0xFF8F3900, 7517},  
02888 {"?", 0xFF663D2E, 7518},  
02889 {"?", 0xFF423500, 7519},  
02890 {"?", 0xFFFFFD6CF, 7520},  
02891 {"?", 0FFE6ACB8, 7521},  
02892 {"?", 0xFFD68196, 7522},  
02893 {"?", 0xFFCC7A85, 7523},  
02894 {"?", 0FFBA544A, 7524},  
02895 {"?", 0FFB36259, 7525},  
02896 {"?", 0FFA63A00, 7526},  
02897 {"?", 0FFEDE8DF, 7527},  
02898 {"?", 0FFE6DFCF, 7528},  
02899 {"?", 0FFD4CBBA, 7529},  
02900 {"?", 0FFADA089, 7530},  
02901 {"?", 0FF80735D, 7531},  
02902 {"?", 0FF594A2D, 7532},  
02903 {"?", 0FF261E06, 7533},
```

```

02904 {"?", 0xFFE6E1D3, 7534},
02905 {"?", 0xFFCCC6AD, 7535},
02906 {"?", 0xFFADA687, 7536},
02907 {"?", 0xFFC6CCB8, 7537},
02908 {"?", 0xFFA2B39B, 7538},
02909 {"?", 0xFFA0A395, 7539},
02910 {"?", 0xFF474747, 7540},
02911 {"?", 0xFFEDF2F2, 7541},
02912 {"?", 0xFFC1D6D0, 7542},
02913 {"?", 0xFFA6B3B3, 7543},
02914 {"?", 0xFF8A9799, 7544},
02915 {"?", 0xFF495C5E, 7545},
02916 {"?", 0xFF304547, 7546},
02917 {"?", 0xFF0A0F0F, 7547},
02918 {"END", 0, -1}
02919 };
02920
02921 /* Based on the manufacturer table at
02922 * https://help.brother-usa.com/app/answers/detail/a_id/75245/~/thread-color-conversion-chart
02923 * using a colour picker.
02924 */
02925 thread_color RobisonAnton_Polyester_codes[] = {
02926 {"Black", 0xF000000, 5596},
02927 {"White", 0xFFFFFFFF, 5597},
02928 /* unfinished */
02929 {"END", 0, -1}
02930 };
02931
02932 thread_color RobisonAnton_Rayon_codes[] = {
02933 {"Light Pink", 0xFFEFCCCE, 2243},
02934 {"Pink", 0xFFFFCFC9, 2223},
02935 {"Pink Bazaar", 0xFFEFC6D3, 2599},
02936 {"Pink Mist", 0xFFFF9B2B7, 2373},
02937 {"Emily Pink", 0xFFFF9AFAD, 2374},
02938 {"Rose", 0xFFFFC9BB2, 2293},
02939 {"Rose Cerise", 0xFFFFC8C99, 2244},
02940 {"Carnation", 0xFFFF2AFC1, 2237},
02941 {"Shrimp", 0xFFFFE5566D, 2246},
02942 {"Bashful Pink", 0xFFFF4476B, 2248},
02943 {"Begonia", 0xFFFF5566D, 2228}, /* TODO: duplicate case value */
02944 {"Azalea", 0xFFFF9848E, 2412},
02945 {"Dusty Rose", 0xFFFF26877, 2375},
02946 {"Rose Tint", 0xFFD8899B, 2591},
02947 {"Burgundy", 0xFF8C2633, 2249},
02948 {"TH Burgundy", 0xFF7C2128, 2608},
02949 {"Russet", 0xFF7A2638, 2252},
02950 {"Pro Firebrand", 0xFF75263D, 2622},
02951 {"Wine", 0xFFFF772D35, 2225},
02952 {"Intense Maroon", 0xFF593344, 2587},
02953 {"Dark Maroon", 0xFF4F213A, 2376},
02954 {"Carbernet", 0xFF931638, 2494},
02955 {"Mountain Rose", 0xFFFF9B2B7, 2495}, /* TODO: duplicate case value */
02956 {"Warm Wine", 0xFF661E2B, 2496},
02957 {"Primrose", 0xFFFF5566D, 2491}, /* TODO: duplicate case value */
02958 {"Perfect Ruby", 0xFF8E2344, 2497},
02959 {"Brushed Burgundy", 0xFFFF6D213F, 2498},
02960 {"Passion Rose", 0xFFFF8E2344, 2499}, /* TODO: duplicate case value */
02961 {"New Berry", 0xFFAD0075, 2500},
02962 {"Petal Pink", 0xFFFFCC9C6, 2501},
02963 {"Memphis Belle", 0xFFFF4BF01, 2502},
02964 {"Desert Bloom", 0xFFFF7BFBF, 2503},
02965 {"Wild Pink", 0xFFCE007C, 2259},
02966 {"Floral Pink", 0xFFED72AA, 2415},
02967 {"Hot Pink", 0xFFD36B9E, 2260},
02968 {"Crimson", 0xFFD60270, 2416},
02969 {"Ruby Glint", 0xFFE22882, 2261},
02970 {"Cherrystone", 0xFFFFAA004F, 2504},
02971 {"Cherry Punch", 0xFFFFAA004F, 2417}, /* TODO: duplicate case value */
02972 {"Cherry Blossom", 0xFFFFEA0F6B, 2262},
02973 {"Red Berry", 0xFFAFA1E2D, 2418},
02974 {"Jockey Red", 0xFFBF0A30, 2281},
02975 {"Very Red", 0xFFBF0A30, 2419}, /* TODO: duplicate case value */
02976 {"Red Berry", 0xFFBF0A30, 2378}, /* TODO: duplicate case value */
02977 {"Foxy Red", 0xFFCE1126, 2263},
02978 {"Tuxedo Red", 0xFFD62828, 2420},
02979 {"Lipstick", 0xFFBF0A30, 2233}, /* TODO: duplicate case value */
02980 {"Scarlet", 0xFFC41E3A, 2219},
02981 {"Radiant Red", 0xFFC41E3A, 2266}, /* TODO: duplicate case value */
02982 {"Wildfire", 0xFFFFA32638, 2267},
02983 {"Carolina Red", 0xFF8C2633, 2268}, /* TODO: duplicate case value */
02984 {"Red Jubilee", 0xFF75263D, 2421}, /* TODO: duplicate case value */
02985 {"Cranberry", 0xFF992135, 2270},
02986 {"Antique Red", 0xFFFFA32638, 2505}, /* TODO: duplicate case value */
02987 {"Devil Red", 0xFFC41E3A, 2506}, /* TODO: duplicate case value */
02988 {"Rosewood", 0xFFFFD81C3F, 2508},
02989 {"Bitterroot", 0xFFFFCE572, 2509},
02990 {"Bisque", 0xFFFF2C4AF, 2377},

```

```
02991 {"Flesh", 0xFFFF4CCAA, 2413},  
02992 {"Flesh Pink", 0xFFFF9BAAA, 2253},  
02993 {"Opal Mist", 0xFFFF7BFBF, 2255}, /* TODO: duplicate case value */  
02994 {"Candy Apple Red", 0xFFAF003D, 2507},  
02995 {"Tawny", 0xFFFF9BF9E, 2256},  
02996 {"Peach", 0xFFFF9C6AA, 2257},  
02997 {"Melon", 0xFFFF98E6D, 2294},  
02998 {"Flamingo", 0xFFFF9A58C, 2258},  
02999 {"Coral", 0xFFFF9872, 2414},  
03000 {"Persimmon", 0xFFE43F4F, 2277},  
03001 {"Peach Blossom", 0xFFFF9C6AA, 2510}, /* TODO: duplicate case value */  
03002 {"Illusion", 0xFFFF9BAAA, 2511}, /* TODO: duplicate case value */  
03003 {"Melonade", 0xFFFF9872, 2512}, /* TODO: duplicate case value */  
03004 {"Honeysuckle", 0xFFE23D28, 2513},  
03005 {"Brite Jade", 0xFF008C82, 2514},  
03006 {"Bluestone", 0xFF00B2A0, 2515},  
03007 {"Aqua Pearl", 0xFF47D6C1, 2516},  
03008 {"Seafrost", 0xFF87DDD1, 2517},  
03009 {"J. Turquoise", 0xFF008789, 2492},  
03010 {"Indian Ocean Blue", 0xFF2DC6D6, 2518},  
03011 {"Surf Blue", 0xFF00A5DB, 2519},  
03012 {"Mid Windsor", 0xFF00A3DD, 2520},  
03013 {"Deep Windsor", 0xFF003F54, 2589},  
03014 {"Pro Dark Blue", 0xFF002D47, 2620},  
03015 {"Mallard Blue", 0xFF006D75, 2521},  
03016 {"Sky Blue", 0xFF9BC4E2, 2239},  
03017 {"Lake Blue", 0xFFAFBCDB, 2304},  
03018 {"Pro Lusty Blue", 0xFF5B77CC, 2614},  
03019 {"Slate Blue", 0xFF6D87A8, 2275},  
03020 {"Blue Frost", 0xFF99D6DD, 2305},  
03021 {"Periwinkle", 0xFF28C4D8, 2306},  
03022 {"Aquamarine", 0xFF00ADC6, 2307},  
03023 {"California Blue", 0xFF00A0C4, 2389},  
03024 {"Baltic Blue", 0xFF008ED6, 2441},  
03025 {"Solar Blue", 0xFF0054A0, 2442},  
03026 {"Pacific Blue", 0xFF008ED6, 2388}, /* TODO: duplicate case value */  
03027 {"Boo Boo Blue", 0xFF00A3DD, 2730}, /* TODO: duplicate case value */  
03028 {"Pro Band Blue", 0xFF00709E, 2737},  
03029 {"Pro Peacock", 0xFF007AA5, 2740},  
03030 {"Light Blue", 0xFFC9E8DD, 2222},  
03031 {"Royal", 0xFF1E1C77, 2210},  
03032 {"Blue Suede", 0xFF002B7F, 2438},  
03033 {"Imperial Blue", 0xFF335687, 2302},  
03034 {"Bridgeport Blue", 0xFFAFBCDB, 2522}, /* TODO: duplicate case value */  
03035 {"China Blue", 0xFF335687, 2523}, /* TODO: duplicate case value */  
03036 {"Pro Imperial", 0xFF26547C, 2612},  
03037 {"Country Blue", 0xFF7796B2, 2524},  
03038 {"Heron Blue", 0xFFC1C9DD, 2525},  
03039 {"Pro Saxon", 0xFF7796B2, 2624}, /* TODO: duplicate case value */  
03040 {"Bright Blue", 0xFF6689CC, 2526},  
03041 {"Soldier Blue", 0xFF5960A8, 2527},  
03042 {"Pro Brilliance", 0xFF0051BA, 2619},  
03043 {"Atlantis Blue", 0xFF6689CC, 2528}, /* TODO: duplicate case value */  
03044 {"Dolphin Blue", 0xFF3A75C4, 2529},  
03045 {"Caribbean Blue", 0xFF75AADB, 2530},  
03046 {"Dana Blue", 0xFFC4D8E2, 2531},  
03047 {"Cadet Blue", 0xFFC4D8E2, 2532}, /* TODO: duplicate case value */  
03048 {"Ozone", 0xFF60AFDD, 2533},  
03049 {"Salem Blue", 0xFF003D6B, 2534},  
03050 {"Blue Ribbon", 0xFF0F2B5B, 2439},  
03051 {"Pro Navy", 0xFF192168, 2625},  
03052 {"Blue Ink", 0xFF2B265B, 2440},  
03053 {"Pro Midnight", 0xFF002654, 2613},  
03054 {"Pro College Blue", 0xFF002649, 2647},  
03055 {"Light Midnight", 0xFF353F5B, 2386},  
03056 {"Fleet Blue", 0xFF35264F, 2450},  
03057 {"Light Navy", 0xFF112151, 2303},  
03058 {"Flag Blue", 0xFF14213D, 2603},  
03059 {"TH Navy", 0xFF002654, 2609}, /* TODO: duplicate case value */  
03060 {"Navy", 0xFF14213D, 2215}, /* TODO: duplicate case value */  
03061 {"Midnight Navy", 0xFF14213D, 2387}, /* TODO: duplicate case value */  
03062 {"Pastel Blue", 0xFFC1C9DD, 2382},  
03063 {"Blue Hint", 0xFFC1C9DD, 2598}, /* TODO: duplicate case value */  
03064 {"Ice Blue", 0xFFB5D1E8, 2300},  
03065 {"Paris Blue", 0xFFD1CEDD, 2283},  
03066 {"Baby Blue", 0xFF99BADD, 2206},  
03067 {"Sun Blue", 0xFFAFBCDB, 2269}, /* TODO: duplicate case value */  
03068 {"Cristy Blue", 0xFFA5BAE0, 2383},  
03069 {"Ultra Blue", 0xFF75B2DD, 2433},  
03070 {"Tropic Blue", 0xFF75AADB, 2434}, /* TODO: duplicate case value */  
03071 {"Blue Horizon", 0xFF6689CC, 2435}, /* TODO: duplicate case value */  
03072 {"Oriental Blue", 0xFF7F8CBF, 2301},  
03073 {"Copen", 0xFF6D87A8, 2245}, /* TODO: duplicate case value */  
03074 {"Jay Blue", 0xFF5B77CC, 2384}, /* TODO: duplicate case value */  
03075 {"Blue", 0xFF2D338E, 2220},  
03076 {"Sapphire", 0xFF2D338E, 2280}, /* TODO: duplicate case value */  
03077 {"Pro Royal", 0xFF00337F, 2627},
```

```

03078 {"Fire Blue", 0xFF003893, 2436},
03079 {"Jamie Blue", 0xFF2D338E, 2385}, /* TODO: duplicate case value */
03080 {"Empire Blue", 0xFF3F2893, 2437},
03081 {"Enchanted Sea", 0xFF3A564F, 2535},
03082 {"Pro Twinkle", 0xFF8499A5, 2617},
03083 {"Rockport Blue", 0xFF9BAAFF, 2536},
03084 {"Wonder Blue", 0xFF5E99AA, 2577},
03085 {"Traditional Gray", 0xFFFFADAFAA, 2540},
03086 {"Steel", 0xFFBFBAFF, 2537},
03087 {"Pro Pearl", 0xFFBAB7AF, 2741},
03088 {"Pro Cool Gray", 0xFFC4C1BA, 2733},
03089 {"Stainless Steel", 0xFFCC1B2, 2538},
03090 {"Chrome", 0xFFFFD1CCBF, 2539},
03091 {"Pro Night Sky", 0xFFADAFAA, 2618}, /* TODO: duplicate case value */
03092 {"Gull", 0xFF8C8984, 2731},
03093 {"Mineral", 0xFF686663, 2729},
03094 {"Black Chrome", 0xFF443D38, 2541},
03095 {"Heather", 0xFFFFDC6C4, 2271},
03096 {"Grape", 0xFFD3B7A3, 2272},
03097 {"Satin Wine", 0xFFB5939B, 2314},
03098 {"Ducky Mauve", 0xFF8E6877, 2422},
03099 {"Pale Orchid", 0xFFFFCBFC9, 2423}, /* TODO: duplicate case value */
03100 {"Orchid", 0xFFE5BFC6, 2379},
03101 {"Lavender", 0xFFEFC6D3, 2276}, /* TODO: duplicate case value */
03102 {"Violet", 0xFFEDC4DD, 2285},
03103 {"Cachet", 0xFFF858CB2, 2424},
03104 {"Tulip", 0xFFC9ADD8, 2286},
03105 {"Mid Lilac", 0xFFE29ED6, 2588},
03106 {"Port Wine", 0xFF512654, 2600},
03107 {"Pro Maroon", 0xFF512D44, 2616},
03108 {"Laurie Lilac", 0xFF8E47AD, 2425},
03109 {"Iris", 0xFFAF72C1, 2288},
03110 {"Raspberry", 0xFF9B4F96, 2426},
03111 {"Mulberry", 0xFF66116D, 2380},
03112 {"Plum Wine", 0xFF63305E, 2490},
03113 {"Purple Twist", 0xFF1E1C77, 2429}, /* TODO: duplicate case value */
03114 {"Violet Blue", 0xFF332875, 2427},
03115 {"Purple Maze", 0xFF35006D, 2428},
03116 {"Pro Brite Star", 0xFF2B1166, 2736},
03117 {"Pro Violet", 0xFF38197A, 2742},
03118 {"Pro Purple", 0xFF35006D, 2628}, /* TODO: duplicate case value */
03119 {"Purple", 0xFF8E47AD, 2254}, /* TODO: duplicate case value */
03120 {"Purple Shadow", 0xFF5B027A, 2430},
03121 {"Dark Purple", 0xFF4C145E, 2381},
03122 {"Mauve", 0xFF8977BA, 2287},
03123 {"Purple Accent", 0xFF44235E, 2431},
03124 {"Hot Peony", 0xFFAA0066, 2590},
03125 {"Passion", 0xFFCE007C, 2291}, /* TODO: duplicate case value */
03126 {"Strawberry", 0xFFAA004F, 2432}, /* TODO: duplicate case value */
03127 {"Plum", 0xFF9E2387, 2292},
03128 {"Misty", 0xFF5E99AA, 2308}, /* TODO: duplicate case value */
03129 {"Mystic Teal", 0xFF609191, 2443},
03130 {"Teal", 0xFF609191, 2309}, /* TODO: duplicate case value */
03131 {"Dark Teal", 0xFF003F54, 2444}, /* TODO: duplicate case value */
03132 {"Mint Julep", 0xFF93DDDB, 2310},
03133 {"Turquoise", 0xFF7FD6DB, 2204},
03134 {"M.D. Green", 0xFF007272, 2445},
03135 {"Seafoam", 0xFF70CE9B, 2311},
03136 {"Isle Green", 0xFF70CE9B, 2312}, /* TODO: duplicate case value */
03137 {"Peppermint", 0xFF35C4AF, 2390},
03138 {"Oceanic Green", 0xFF006D75, 2446}, /* TODO: duplicate case value */
03139 {"Pro Teal", 0xFF006B77, 2621},
03140 {"Garden Green", 0xFF006663, 2447},
03141 {"Pro Green", 0xFF006D66, 2735},
03142 {"Pine Green", 0xFF008C82, 2391}, /* TODO: duplicate case value */
03143 {"Greenstone", 0xFF008272, 2448},
03144 {"Fern Green", 0xFF006663, 2449}, /* TODO: duplicate case value */
03145 {"Pro Hunter", 0xFF006D66, 2615}, /* TODO: duplicate case value */
03146 {"Palm Leaf", 0xFFBCC1B2, 2241},
03147 {"Flite Green", 0xFFC6D6A0, 2282},
03148 {"Willow", 0xFFFF9EAA99, 2221},
03149 {"Sprite", 0xFFFFB2D8D8, 2313},
03150 {"Moss", 0xFF7AA891, 2278},
03151 {"Wintergreen", 0xFF7AA891, 2594}, /* TODO: duplicate case value */
03152 {"Green Forest", 0xFF006056, 2451},
03153 {"Pro Forest", 0xFF006056, 2743}, /* TODO: duplicate case value */
03154 {"Harbor Green", 0xFF4F6D5E, 2392},
03155 {"Evergreen", 0xFF024930, 2315},
03156 {"Pro Dark Green", 0xFF004438, 2734},
03157 {"Lizzy Lime", 0xFF284C3F, 2631},
03158 {"D.H. Green", 0xFF282D26, 2411},
03159 {"Celery", 0xFFFF2EABC, 2316},
03160 {"Pistachio", 0xFFCC693, 2250},
03161 {"Olive Drab", 0xFF5E663A, 2317},
03162 {"Olive", 0xFF779182, 2202},
03163 {"Pale Green", 0xFFC9E8DD, 2318}, /* TODO: duplicate case value */
03164 {"Green Pearl", 0xFF93DDDB, 2452}, /* TODO: duplicate case value */

```

```
03165 {"Sea Mist", 0xFF93DDDB, 2393}, /* TODO: duplicate case value */
03166 {"Mint", 0xFFB5E8BF, 2238},
03167 {"Spruce", 0xFFAADD6D, 2279},
03168 {"Nile", 0xFFA0DB8E, 2211},
03169 {"Green Oak", 0xFFB5CC8E, 2319},
03170 {"Erin Green", 0xFF8CD600, 2320},
03171 {"Pro Erin", 0xFF56AA1C, 2738},
03172 {"Emerald", 0xFF339E35, 2214},
03173 {"Dark Emerald", 0xFF007A3D, 2453},
03174 {"Light Kelly", 0xFF007A3D, 2410}, /* TODO: duplicate case value */
03175 {"Kelly", 0xFF007A3D, 2240}, /* TODO: duplicate case value */
03176 {"Dark Green", 0xFF008751, 2208},
03177 {"Fleece Green", 0xFF006B3F, 2454},
03178 {"TH Green", 0xFF006854, 2607},
03179 {"Harvest Green", 0xFF1EB53A, 2578},
03180 {"Vibrant Green", 0xFF009E49, 2579},
03181 {"Green Grass", 0xFF009E49, 2580}, /* TODO: duplicate case value */
03182 {"Deep Green", 0xFF006B3F, 2284}, /* TODO: duplicate case value */
03183 {"Green Bay", 0xFF006B54, 2455},
03184 {"Jungle Green", 0xFF007C66, 2597},
03185 {"Peapod", 0xFFFFA3AF07, 2456},
03186 {"Pastoral Green", 0xFF7FBA00, 2321},
03187 {"Green Dust", 0xFF7FBA00, 2457}, /* TODO: duplicate case value */
03188 {"Ming", 0xFF7FBA00, 2322}, /* TODO: duplicate case value */
03189 {"Meadow", 0xFF568E14, 2226},
03190 {"Tamarack", 0xFF939905, 2230},
03191 {"Palmetto", 0xFF566314, 2229},
03192 {"Green Petal", 0xFF024930, 2458}, /* TODO: duplicate case value */
03193 {"Sage", 0xFF547730, 2595},
03194 {"Hedge", 0xFF3F4926, 2601},
03195 {"Green", 0xFF3A7728, 2209},
03196 {"Green Sail", 0xFF193833, 2459},
03197 {"Holly", 0xFF215B33, 2323},
03198 {"Field Green", 0xFF265142, 2460},
03199 {"Dress Green", 0xFF3F4926, 2584}, /* TODO: duplicate case value */
03200 {"Foliage Green", 0xFF99840A, 2542},
03201 {"Autumn Green", 0xFFA38205, 2543},
03202 {"Desert Cactus", 0xFF897719, 2544},
03203 {"Cypress", 0xFF707014, 2545},
03204 {"Crescent Moon", 0xFF848205, 2546},
03205 {"Pebblestone", 0xFFE2E584, 2547},
03206 {"Sun Shadow", 0xFF998E07, 2548},
03207 {"Blue Spruce", 0xFF00494F, 2549},
03208 {"Newport", 0xFF4F6D5E, 2550}, /* TODO: duplicate case value */
03209 {"Spring Garden", 0xFF779182, 2551}, /* TODO: duplicate case value */
03210 {"Water Lilly", 0xFF546856, 2554},
03211 {"Ivy", 0xFF0C3026, 2552},
03212 {"Dark Army Green", 0xFF233A2D, 2553},
03213 {"Army Green", 0xFF213D30, 2728},
03214 {"Pastel Green", 0xFFC9D6A3, 2555},
03215 {"Pollen Gold", 0xFFE0AA0F, 2556},
03216 {"Pale Yellow", 0xFFFF4E287, 2557},
03217 {"Buttercup", 0xFFFFC61E, 2558},
03218 {"Tusk", 0xFFFF7E8AA, 2559},
03219 {"Moonbeam", 0xFFF9DD16, 2560},
03220 {"Black Eyed Susie", 0xFFC6A00C, 2561},
03221 {"Bullion", 0xFFA37F14, 2562},
03222 {"Chinese Yellow", 0xFFFF7E8AA, 2324}, /* TODO: duplicate case value */
03223 {"Maize", 0xFFFF7E8AA, 2264}, /* TODO: duplicate case value */
03224 {"Wheat", 0xFFEADD96, 2461},
03225 {"Pro Maize", 0xFFF9E08C, 2732},
03226 {"Glow", 0xFFFFD87F, 2234},
03227 {"Star Gold", 0xFFFFCD856, 2408},
03228 {"Mango", 0xFFFFCA311, 2394},
03229 {"Yellow Mist", 0xFFFFCA311, 2409}, /* TODO: duplicate case value */
03230 {"Yellow", 0xFFFFCA311, 2213}, /* TODO: duplicate case value */
03231 {"Sunflower", 0xFFF9DD16, 2462}, /* TODO: duplicate case value */
03232 {"Lemon", 0xFFFF4ED47, 2325},
03233 {"Daffodil", 0xFFFF9E814, 2326},
03234 {"Merit Gold", 0xFFFFCB514, 2463},
03235 {"Cornsilk", 0xFFFFFCCC49, 2395},
03236 {"Nectar", 0xFFFFFC61E, 2464}, /* TODO: duplicate case value */
03237 {"Scholastic", 0xFFFFCBF49, 2465},
03238 {"Canary Yellow", 0xFFFFCE016, 2235},
03239 {"Pro Gold", 0xFFFFFC61E, 2626}, /* TODO: duplicate case value */
03240 {"Manila", 0xFFFFCD116, 2466},
03241 {"Goldenrod", 0xFFFFCD116, 2242}, /* TODO: duplicate case value */
03242 {"Brite Yellow", 0xFFFFFCCC49, 2396}, /* TODO: duplicate case value */
03243 {"Honeydew", 0xFFFF99B0C, 2327},
03244 {"Pumpkin", 0xFFFF77F00, 2328},
03245 {"Orangeade", 0xFFFF74902, 2467},
03246 {"Sun Orange", 0xFFFF74902, 2397}, /* TODO: duplicate case value */
03247 {"Paprika", 0xFFFF95602, 2236},
03248 {"Saffron", 0xFFFF93F26, 2329},
03249 {"Tex Orange", 0xFFFF95602, 2468}, /* TODO: duplicate case value */
03250 {"Orange", 0xFFFF96B07, 2218},
03251 {"Dark Tex Orange", 0xFFFF96302, 2469},
```

```

03252 {"Old Dark Tex Orange", 0xFFA53F0F, 2581},
03253 {"Golden Poppy", 0xFFFFC8744, 2330},
03254 {"Rust", 0xFFBC4F07, 2289},
03255 {"Copper", 0xFFFFAF7505, 2295},
03256 {"Light Bronze", 0xFFC18E60, 2493},
03257 {"Visor Gold", 0xFFFFCE87, 2398},
03258 {"Goldenlite", 0xFFFFFCC49, 2605}, /* TODO: duplicate case value */
03259 {"Honey", 0xFFFFCBA5E, 2247},
03260 {"Marigold", 0xFFFFFCC49, 2216}, /* TODO: duplicate case value */
03261 {"Mustard", 0xFFE0AA0F, 2331}, /* TODO: duplicate case value */
03262 {"Sun Gold", 0xFFE0AA0F, 2212}, /* TODO: duplicate case value */
03263 {"Karat", 0xFE0AA0F, 2470}, /* TODO: duplicate case value */
03264 {"Penny", 0xFF2BF49, 2332},
03265 {"New Gold", 0xFFFCCE87, 2399}, /* TODO: duplicate case value */
03266 {"Pro Beige", 0xFFE2D6B5, 2630},
03267 {"Marine Gold", 0xFFC6A00C, 2596}, /* TODO: duplicate case value */
03268 {"Ginger", 0xFFA37F14, 2333}, /* TODO: duplicate case value */
03269 {"Shimmering Gold", 0xFC6A00C, 2471}, /* TODO: duplicate case value */
03270 {"Old Gold", 0xFFBF910C, 2201},
03271 {"Salmon", 0xFFEAEB2B2, 2299},
03272 {"Dark Rust", 0xFFC13828, 2205},
03273 {"Terra Cotta", 0xFFA03033, 2334},
03274 {"Pro Red", 0xFFAF1E2D, 2623}, /* TODO: duplicate case value */
03275 {"Auburn", 0xFFAF1E2D, 2472}, /* TODO: duplicate case value */
03276 {"Bone", 0xFFFFAE6CC, 25821},
03277 {"Ivory", 0xFFFFAE6CF, 2335},
03278 {"Opaline", 0xFFF7D3B5, 2473},
03279 {"Ecru", 0xFFEDD3BC, 2232},
03280 {"Wicker", 0xFFAA753F, 2489},
03281 {"Tan", 0xFFD3A87C, 2273},
03282 {"Cottage Beige", 0xFFEDD3B5, 2593},
03283 {"Rattan", 0xFFC1A875, 2474},
03284 {"Gold", 0FFE2BF9B, 2203},
03285 {"Mocha Cream", 0xFFD3A87C, 2475}, /* TODO: duplicate case value */
03286 {"Topaz", 0xFFBF910C, 2400}, /* TODO: duplicate case value */
03287 {"Ashley Gold", 0xFFD18E54, 2401},
03288 {"Amber Beige", 0xFFD8B596, 2336},
03289 {"Seashell", 0xFFD6CCAF, 2476},
03290 {"Light Maize", 0xFFFF2E3C4, 2604},
03291 {"Beige", 0xFFAA753F, 2224}, /* TODO: duplicate case value */
03292 {"Sand Dune", 0xFF6B4714, 2477},
03293 {"Taupe", 0xFD1BF91, 2298},
03294 {"Pro Brown", 0xFF6C463D, 2610},
03295 {"Chocolate", 0xFF876028, 2227},
03296 {"Pro Walnut", 0xFF755426, 2629},
03297 {"Light Cocoa", 0xFF755426, 2478}, /* TODO: duplicate case value */
03298 {"Cocoa Mulch", 0xFFB28260, 2488},
03299 {"Brown", 0xF593D2B, 2251},
03300 {"Dark Brown", 0xF593D2B, 2372}, /* TODO: duplicate case value */
03301 {"Espresso", 0xFF3F302B, 2337},
03302 {"Bamboo", 0xFFFFC18E60, 2338}, /* TODO: duplicate case value */
03303 {"Almond", 0xFFAF7505, 2479}, /* TODO: duplicate case value */
03304 {"Toast", 0xFFBA7530, 2231},
03305 {"Sienna", 0xFF755426, 2402}, /* TODO: duplicate case value */
03306 {"K.A. Bronze", 0xFFB26B70, 2480},
03307 {"Pro Cinnamon", 0xFFA2464E, 2611},
03308 {"Date", 0xFF9B4F19, 2290},
03309 {"Hazel", 0xFF9B4F19, 2481}, /* TODO: duplicate case value */
03310 {"Coffee Bean", 0xFF5B2D28, 2339},
03311 {"Dogwood", 0xFF593D2B, 2563}, /* TODO: duplicate case value */
03312 {"Mahogany", 0xFF3F302B, 2564}, /* TODO: duplicate case value */
03313 {"Best Brown", 0xFF3D3028, 2566},
03314 {"Mushroom", 0xFF633A11, 2567},
03315 {"Perfect Tan", 0xFFC1A875, 2568}, /* TODO: duplicate case value */
03316 {"Earthen Tan", 0xFF7A5B11, 2569},
03317 {"Golden Tan", 0xFFC1A875, 2570}, /* TODO: duplicate case value */
03318 {"14 Kt. Gold", 0FFF2BF49, 2586}, /* TODO: duplicate case value */
03319 {"TH Gold", 0FFF2CE68, 2606},
03320 {"24 Kt. Gold", 0FFD88C02, 2602},
03321 {"Platinum", 0FFC1B5A5, 2571},
03322 {"Pro Gray", 0FF99897C, 2739},
03323 {"Grayrod", 0FFADA07A, 2572},
03324 {"Pewter", 0FFADA07A, 2573}, /* TODO: duplicate case value */
03325 {"Aspen White", 0FFF5E3CC, 2574},
03326 {"Dark Taupe", 0FF66594C, 2575},
03327 {"Egyptian Brown", 0FF493533, 2576},
03328 {"Oyster", 0FFF5EBE0, 2403},
03329 {"Gray", 0FFDDC6C4, 2207}, /* TODO: duplicate case value */
03330 {"Pearl Gray", 0FFDBD3D3, 2340},
03331 {"Steel Gray", 0FFD8CCD1, 2274},
03332 {"Skylight", 0FFCCC1C6, 2482},
03333 {"Cloud", 0FFAFAAA3, 2483},
03334 {"Silver Steel", 0FFADAFAA, 2592}, /* TODO: duplicate case value */
03335 {"Banner Gray", 0FF919693, 2585},
03336 {"Silvery Gray", 0FF8C8984, 2484}, /* TODO: duplicate case value */
03337 {"Cinder", 0FFCCC1C6, 2404}, /* TODO: duplicate case value */
03338 {"Saturn Gray", 0FFDBD3D3, 2485}, /* TODO: duplicate case value */

```

```
03339 {"Dover Gray", 0xFFCCC1C6, 2405}, /* TODO: duplicate case value */
03340 {"Storm Gray", 0xFFB2A8B5, 2486},
03341 {"Sterling", 0xFFA893AD, 2406},
03342 {"Metal", 0xFF666D70, 2407},
03343 {"Twilight", 0xFF686663, 2217}, /* TODO: duplicate case value */
03344 {"Aged Charcoal", 0xFF443D38, 2565}, /* TODO: duplicate case value */
03345 {"Charcoal", 0xFF777772, 2265},
03346 {"Smokey", 0xFFFF353842, 2487},
03347 {"Ash", 0xFF3A4972, 2341},
03348 {"Black", 0xFF1C2630, 2296},
03349 {"Snow White", 0xFFFF5EBE0, 2297}, /* TODO: duplicate case value */
03350 {"Natural White", 0xFFFF5EDDE, 2342},
03351 {"Eggshell", 0xFFFF0E8D6, 2343},
03352 {"Jet Black", 0xFF1C2630, 2632}, /* TODO: duplicate case value */
03353 {"END", 0, -1}
03354 };
03355
03356 thread_color Sigma_Polyester_codes[] = {
03357 {"White", 0xFFFFFFFF, 10},
03358 {"Black", 0x00000000, 20},
03359 {"Light Neon Green", 0xFFFFEDFF50, 21},
03360 {"Neon Green", 0xFF96E845, 32},
03361 {"Light Neon Orange", 0xFFFFE756, 33},
03362 {"Med Neon Orange", 0xFFFF7824, 43},
03363 {"Neon Pink", 0xFFFF28DA6, 46},
03364 {"Neon Orange Pink", 0xFFC70C57, 47},
03365 {"Silver", 0xFFE22D2A, 101},
03366 {"Silver Diamond", 0xFFB8B8B8, 102},
03367 {"Lava Stone", 0xFF889186, 112},
03368 {"Medium Grey", 0xFF737F7F, 115},
03369 {"Dark Platinum", 0xFF565E5A, 116},
03370 {"Charcoal", 0xFF515250, 117},
03371 {"Badger Grey", 0xFF787668, 118},
03372 {"Pumpkin Orange", 0xFFED572F, 135},
03373 {"Turquoise", 0xFF2EA59C, 138},
03374 {"Dark Wedgewood", 0xFF396276, 142},
03375 {"Cardinal Red", 0xFF9B3B40, 213},
03376 {"Maroon", 0xFFFF6C3E47, 216},
03377 {"Rust", 0xFFBAE4D, 253},
03378 {"Medium Rust", 0xFFBB3D2E, 255},
03379 {"Natural Pink", 0xFFFF9DFCF, 301},
03380 {"Baby Pink", 0xFFFFBDDED6, 303},
03381 {"Piggy Pink", 0xFFFF7CD5D5, 304},
03382 {"Sweet Pink", 0xFFFF2AFB4, 305},
03383 {"Blushing Pink", 0xFFE8418C, 307},
03384 {"Pink", 0xFFE77F9D, 309},
03385 {"Rose Pink", 0xFFF06F8C, 313},
03386 {"Green", 0xFF008340, 317},
03387 {"Shocking Pink", 0xFFDF99B6, 321},
03388 {"Ruby", 0xFF820052, 325},
03389 {"Garnet", 0xFFFB1415F, 333},
03390 {"Light Purple", 0xFFC394AE, 345},
03391 {"Medium Purple", 0xFFA86E91, 347},
03392 {"Dark Grape", 0xFF694169, 348},
03393 {"Pastel Light Pink", 0xFFE6CFD5, 376},
03394 {"Light Baby Blue", 0xFFA8BED7, 379},
03395 {"Crystal Blue", 0xFFA0BFD7, 380},
03396 {"Very Light Lavender", 0xFF90A6C6, 381},
03397 {"Cornflower", 0xFF8FAFC6, 382},
03398 {"Lavender", 0xFFB1B8D3, 383},
03399 {"Denim", 0xFF416C9B, 385},
03400 {"Light Violet", 0xFF7D77AF, 386},
03401 {"Misty Rose", 0xFFFFADAF4, 387},
03402 {"Grape", 0xFF664090, 390},
03403 {"Lt. Weathered Blue", 0xFFEAFOF9, 402},
03404 {"Baby Blue", 0xFFA6D8F6, 403},
03405 {"Med Baby Blue", 0xFF7B9CB0, 404},
03406 {"Med Pastel Blue", 0xFF648DC7, 406},
03407 {"Blue Raspberry", 0xFF3D6AA1, 409},
03408 {"Med Royal Blue", 0xFF2D4491, 413},
03409 {"Ocean Blue", 0xFF143D7A, 414},
03410 {"Med Navy", 0xFF113263, 415},
03411 {"Dark Navy", 0xFF0E1F38, 423},
03412 {"Bright Sunshine", 0xFF0E1F38, 432}, /* TODO: duplicate case value */
03413 {"Teal", 0xFF0091A5, 443},
03414 {"Deep Teal", 0xFF005B63, 448},
03415 {"Dark Teal", 0xFF00474D, 449},
03416 {"Old Gold", 0xFFE5B15C, 466},
03417 {"Cream", 0xFFD5BF9B, 501},
03418 {"Pale Salmon", 0xFFFFFD085, 503},
03419 {"Med Peach", 0xFFFF6B08E, 505},
03420 {"Pink Salmon", 0xFFB3E851, 506},
03421 {"Dark Peach", 0xFFFF1A236, 508},
03422 {"Dark Brown", 0xFF6E4337, 513},
03423 {"Pale Red", 0xFFD8493E, 527},
03424 {"Heron Blue", 0xFF697698, 541},
03425 {"Pale Yellow", 0xFFFFDE896, 601},
```

```

03426 {"Pastel Yellow", 0xFFEDE55D, 602},
03427 {"Golden Puppy", 0xFFFFDA200, 609},
03428 {"Buttercup", 0xFFFDE896, 612}, /* TODO: duplicate case value */
03429 {"Treasure Gold", 0xFFCEB24C, 616},
03430 {"Old Gold", 0xFFAD953E, 619},
03431 {"Pale Apricot", 0xFFFFEF9EA, 627},
03432 {"Tan", 0xFFBD9565, 628},
03433 {"Mellow Yellow", 0xFFFFDF76C, 632},
03434 {"Lemon", 0xFFEDEF05, 633},
03435 {"Amber", 0xFFFF8C300, 646},
03436 {"Mandarina", 0xFFE77817, 649},
03437 {"Orange", 0xFFFFE66535, 650},
03438 {"Golden Rod", 0xFFC69632, 652},
03439 {"Light Olive", 0xFF98996D, 653},
03440 {"Bright Gold", 0xFFC98300, 654},
03441 {"Blue-Green", 0xFF007B8D, 688},
03442 {"Forrest Green", 0xFF004D3D, 695},
03443 {"Midnight Blue", 0xFF007EBA, 697},
03444 {"Med Red", 0xFFCF0040, 700},
03445 {"Med Blue", 0xFF28438C, 809},
03446 {"Sweet Apricot", 0xFFD0B478, 812},
03447 {"Skin", 0xFFE5BE6C, 818},
03448 {"Jade", 0xFF449284, 825},
03449 {"Light Silver", 0xFFFCFCFC, 829},
03450 {"Papaya Whip", 0xFFDC875E, 831},
03451 {"Cooper", 0xFFB4705D, 832},
03452 {"Light Pecan", 0xFF9B5C4B, 833},
03453 {"Burnt Rust", 0xFFA93121, 838},
03454 {"Vegas Gold", 0xFFB18B00, 842},
03455 {"Med Brown", 0xFF86462E, 857},
03456 {"Med Russett", 0xFF614125, 859},
03457 {"Med Copper", 0xFFB25C31, 864},
03458 {"Dark Driftwood", 0xFF806A61, 873},
03459 {"Birch", 0xFF634831, 878},
03460 {"Dark Chocolate", 0xFF1A0C06, 891},
03461 {"Sky Blue 2", 0xFF96D5C8, 903},
03462 {"Aquamarine", 0xFFB4DCD8, 904},
03463 {"Golden Brown", 0xFFAF7D3E, 905},
03464 {"Sea Blue", 0xFF00A3A0, 906},
03465 {"Deep Sea", 0xFF00405D, 913},
03466 {"Pastel Mint", 0xFFC9E3C5, 947},
03467 {"True Green", 0xFF55AF78, 949},
03468 {"Med Olive", 0xFF858325, 951},
03469 {"Olive", 0XF61601C, 955},
03470 {"Light Jade", 0xFF709188, 961},
03471 {"Smith Apple", 0xFFBEDC8C, 984},
03472 {"Light Lime", 0xFFBEE678, 985},
03473 {"Grass Green", 0xFF76C850, 988},
03474 {"Deep Teal", 0XF466E64, 448},
03475 {"Med Forrest Green", 0xFF356936, 992},
03476 {"Deep Violet", 0xFF4B4884, 1031},
03477 {"Light Natural", 0xFFEDEDD2, 1140},
03478 {"Wheat", 0FFF3D8A8, 1145},
03479 {"Desert Sand", 0FFC8BE96, 1148},
03480 {"Egyptian Blue", 0FF243A7D, 1163},
03481 {"Gecko", 0XF86BE4E, 1183},
03482 {"Burgundy", 0FF8E4044, 1241},
03483 {"Med Orchid", 0FF893480, 1323},
03484 {"Med Purple", 0FF8C6DAA, 1324},
03485 {"Very Old Gold", 0FFB6A36C, 1552},
03486 {"Light Spruce", 0FF2E9F76, 1615},
03487 {"Paris Green", 0FF98C173, 1619},
03488 {"Timberwolf", 0FFCDCCD, 1707},
03489 {"Bright Blue", 0FF2A377E, 2031},
03490 {"Turquoise Blue", 0FF006CA5, 2093},
03491 {"Dark Wine", 0FF834455, 2250},
03492 {"Beige", 0FFD0A44F, 2518},
03493 {"Gold", 0FFED9206, 2519},
03494 {"Med Orange", 0FFEDEF05, 3001}, /* TODO: duplicate case value */
03495 {"Dark Salmon", 0FFC07A46, 3014},
03496 {"Fire Red", 0FFB43C3C, 3015},
03497 {"Saddle Brown", 0FF915F46, 3142},
03498 {"Yellow Sun", 0FFFFC500, 4117},
03499 {"Deep Taupe", 0FFA68A68, 4371},
03500 {"Sky Blue", 0FF00A4D9, 4419},
03501 {"Wild Peacock", 0FF0B7F85, 4627},
03502 {"Millard Green", 0FF002D1F, 4735},
03503 {"Dark Blue", 0FF11263C, 5552},
03504 {"Powder Blue", 0FF91B9E2, 5554},
03505 {"Froggy Green", 0FF429648, 5557},
03506 {"Stone Grey", 0FF878C8C, 8010},
03507 {"END", 0, -1}
03508 },
03509
03510 thread_color Sulky_Rayon_codes[] = {
03511 {"Cornsilk", 0FFEFC810, 502},
03512 {"Deep Arctic Sky", 0FF0C082D, 505},

```

```
03513 {"Nutmeg", 0xFFB26C29, 521},  
03514 {"Autumn Gold", 0xFFE79002, 523},  
03515 {"English Green", 0xFF34481E, 525},  
03516 {"Cobalt Blue", 0xFF113675, 526},  
03517 {"Forest Green", 0xFF111408, 538},  
03518 {"Lipstick", 0FFE10000, 561},  
03519 {"Spice", 0xFFFFFB435, 562},  
03520 {"Butterfly Gold", 0xFFF3A001, 567},  
03521 {"Cinnamon", 0xFFE66D00, 568},  
03522 {"Garden Green", 0xFF165F28, 569},  
03523 {"Deep Aqua", 0xFF088E6C, 571},  
03524 {"Blue Ribbon", 0xFF100A7C, 572},  
03525 {"Mint Julep", 0xFF35693D, 580},  
03526 {"Dusty Peach", 0xFFE9BD96, 619},  
03527 {"Sunset", 0xFFCD3900, 621},  
03528 {"Moss Green", 0xFF777113, 630},  
03529 {"Med. Aqua", 0xFF1C6F51, 640},  
03530 {"Arctic Sky", 0xFF262345, 643},  
03531 {"Bright White", 0xFFF9F9FF, 1001},  
03532 {"Soft White", 0xFFF9F9F4, 1002},  
03533 {"Black", 0xF000000, 1005},  
03534 {"Steel Gray", 0xFFB7A9AC, 1011},  
03535 {"Med. Peach", 0xFFE1AF9A, 1015},  
03536 {"Pastel Coral", 0xFFEC968C, 1016},  
03537 {"Pastel Peach", 0xFFEFDFBD, 1017},  
03538 {"Peach", 0xFFECA082, 1019},  
03539 {"Dark Peach", 0xFFFFF08278, 1020},  
03540 {"Maple", 0xFFEB6602, 1021},  
03541 {"Cream", 0xFFFFF7D5, 1022},  
03542 {"Yellow", 0xFFFFE669, 1023},  
03543 {"Goldenrod", 0xFFFFFB800, 1024},  
03544 {"Mine Gold", 0xFFD78000, 1025},  
03545 {"Baby Blue", 0xFFBEC3E1, 1028},  
03546 {"Med. Blue", 0xFFA0C3EB, 1029},  
03547 {"Periwinkle", 0xFFA6A2C6, 1030},  
03548 {"Med. Orchid", 0xFFDFBEC8, 1031},  
03549 {"Med. Purple", 0xFFE68CEB, 1032},  
03550 {"Dk. Orchid", 0xFFD86496, 1033},  
03551 {"Burgundy", 0xFFC6323C, 1034},  
03552 {"Dk. Burgundy", 0xFF790000, 1035},  
03553 {"Lt. Red", 0xFFFF90000, 1037},  
03554 {"True Red", 0xFFEB0000, 1039},  
03555 {"Med. Dk. Khaki", 0xFF877375, 1040},  
03556 {"Med. Dk. Gray", 0xFF8C7F83, 1041},  
03557 {"Bright Navy Blue", 0xFF321E50, 1042},  
03558 {"Dk. Navy", 0xFF190525, 1043},  
03559 {"Midnight Blue", 0xFF1D062F, 1044},  
03560 {"Lt. Teal", 0xFFC3EFBF, 1045},  
03561 {"Teal", 0xFF2E8359, 1046},  
03562 {"Mint Green", 0xFFA6C284, 1047},  
03563 {"Grass Green", 0xFF42A021, 1049},  
03564 {"Xmas Green", 0xFF1E6419, 1051},  
03565 {"Med. Dk. Ecru", 0xFFEEBEAE, 1054},  
03566 {"Tawny Tan", 0xFFEBBC80, 1055},  
03567 {"Med Tawny Tan", 0xFFAF5B00, 1056},  
03568 {"Dk Tawny Tan", 0xFF642702, 1057},  
03569 {"Tawny Brown", 0xFF663500, 1058},  
03570 {"Dk Tawny Brown", 0xFF530601, 1059},  
03571 {"Pale Yellow", 0xFFFFF7B9, 1061},  
03572 {"Pale Yellow-Green", 0xFFFF0F8EC, 1063},  
03573 {"Pale Peach", 0xFFE6B4AA, 1064},  
03574 {"Orange Yellow", 0xFFFFF9100, 1065},  
03575 {"Primrose", 0xFFFFF180, 1066},  
03576 {"Lemon Yellow", 0xFFFFFFF85, 1067},  
03577 {"Pink Tint", 0xFFF3DBD9, 1068},  
03578 {"Gold", 0xFFF6CE69, 1070},  
03579 {"Off White", 0xFFF9F9EA, 1071},  
03580 {"Pale Powder Blue", 0xFFFD6D5E8, 1074},  
03581 {"Royal Blue", 0xFF5A5A8B, 1076},  
03582 {"Jade Tint", 0xFFBECD8, 1077},  
03583 {"Tangerine", 0xFFFFF6600, 1078},  
03584 {"Emerald Green", 0xFF175523, 1079},  
03585 {"Orchid", 0xFFDC82A0, 1080},  
03586 {"Brick", 0xFFF06E78, 1081},  
03587 {"Ecru", 0xFF7E3BB, 1082},  
03588 {"Spark Gold", 0xFFFFFC100, 1083},  
03589 {"Silver", 0xFFE2CFC7, 1085},  
03590 {"Pale Sea Foam", 0xFFFF9F9E0, 1086},  
03591 {"Deep Peacock", 0xFF16625F, 1090},  
03592 {"Med Turquoise", 0xFF26BFCA, 1094},  
03593 {"Turquoise", 0xFF10D1BD, 1095},  
03594 {"Dk Turquoise", 0xFF0F6978, 1096},  
03595 {"Lt Grass Green", 0xFFC2D37D, 1100},  
03596 {"True Green", 0xFF098531, 1101},  
03597 {"Dk Khaki", 0xFF02140F, 1103},  
03598 {"Pastel Yellow-Grn", 0xFFA5AF68, 1104},  
03599 {"Lt Mauve", 0xFFFFAA4A4, 1108},
```

```
03600 {"Hot Pink", 0xFFDC6496, 1109},  
03601 {"Pastel Orchid", 0xFFFFCCBDF, 1111},  
03602 {"Royal Purple", 0xFF46016E, 1112},  
03603 {"Pastel Mauve", 0xFFF0C8B4, 1113},  
03604 {"Lt Pink", 0xFFFF0B9B9, 1115},  
03605 {"Mauve", 0xFFFF5A9A0, 1117},  
03606 {"Dk Mauve", 0xFFB46E75, 1119},  
03607 {"Pale Pink", 0xFFF0D6D2, 1120},  
03608 {"Pink", 0xFFFFAB9CB, 1121},  
03609 {"Purple", 0xFFFF82288E, 1122},  
03610 {"Sun Yellow", 0xFFFFFEC00, 1124},  
03611 {"Tan", 0xFFDC8C17, 1126},  
03612 {"Med Ecru", 0xFFFFAECC6, 1127},  
03613 {"Dk Ecru", 0xFFC39471, 1128},  
03614 {"Brown", 0xFF6A1F06, 1129},  
03615 {"Dark Brown", 0xFF551602, 1130},  
03616 {"Cloister Brown", 0xFF490002, 1131},  
03617 {"Peacock Blue", 0xFF507DAA, 1134},  
03618 {"Pastel Yellow", 0xFFFFF072, 1135},  
03619 {"Yellow Orange", 0xFFFFFBEO0, 1137},  
03620 {"True Blue", 0xFF4A5870, 1143},  
03621 {"Powder Blue", 0xFFB4E1EB, 1145},  
03622 {"Xmas Red", 0xFFEB0000, 1147}, /* TODO: duplicate case value */  
03623 {"Lt Coral", 0xFFFFBDDB, 1148},  
03624 {"Deep Ecru", 0xFFE8C89C, 1149},  
03625 {"Powder Blue Tint", 0xFFE2E2EB, 1151},  
03626 {"Coral", 0xFFFFA9999, 1154},  
03627 {"Lt Army Green", 0xFF636327, 1156},  
03628 {"Dk Maple", 0xFFBA4500, 1158},  
03629 {"Temple Gold", 0xFFD39D00, 1159},  
03630 {"Deep Teal", 0xFF10394A, 1162},  
03631 {"Lt Sky Blue", 0xFFDFE5EB, 1165},  
03632 {"Med Steel Gray", 0xFF8E7E7E, 1166},  
03633 {"Maize Yellow", 0xFFFFD226, 1167},  
03634 {"True Orange", 0xFFFF5740A, 1168},  
03635 {"Bayberry Red", 0xFF9C0000, 1169},  
03636 {"Lt Brown", 0xFF975F2F, 1170},  
03637 {"Weathered Blue", 0xFF08180E, 1171},  
03638 {"Med Weathered Blue", 0xFF6E788C, 1172},  
03639 {"Med Army Green", 0xFF59591D, 1173},  
03640 {"Dk Pine Green", 0xFF0D2904, 1174},  
03641 {"Dk Avocado", 0xFF152D04, 1175},  
03642 {"Med Dk Avocado", 0xFF515308, 1176},  
03643 {"Avocado", 0xFF899812, 1177},  
03644 {"Dk Taupe", 0xFF8F623D, 1179},  
03645 {"Med Taupe", 0xFFA58973, 1180},  
03646 {"Rust", 0xFFFCB000, 1181},  
03647 {"Blue Black", 0xFF020114, 1182},  
03648 {"Black Cherry", 0xFF320614, 1183},  
03649 {"Orange Red", 0xFFFF6600, 1184}, /* TODO: duplicate case value */  
03650 {"Golden Yellow", 0xFFFFCBE05, 1185},  
03651 {"Sable Brown", 0xFF5B0000, 1186},  
03652 {"Mimosa Yellow", 0xFFFFFE500, 1187},  
03653 {"Red Geranium", 0xFFFFF004B, 1188},  
03654 {"Dk Chestnut", 0xFF4B122D, 1189},  
03655 {"Med Burgundy", 0xFFFA04656, 1190},  
03656 {"Dk Rose", 0xFFBD1E60, 1191},  
03657 {"Fuchsia", 0xFFD21E82, 1192},  
03658 {"Lavender", 0FFE6AFD2, 1193},  
03659 {"Lt Purple", 0xFFD274D7, 1194},  
03660 {"Dk Purple", 0xFF370150, 1195},  
03661 {"Blue", 0xFF96C3E1, 1196},  
03662 {"Med Navy", 0xFF220F34, 1197},  
03663 {"Dusty Navy", 0xFF3C5075, 1198},  
03664 {"Admiral Navy Blue", 0xFF2A143F, 1199},  
03665 {"Med Dk Navy", 0xFF140B2D, 1200},  
03666 {"Med Powder Blue", 0xFF648BBE, 1201},  
03667 {"Deep Turquoise", 0xFF182B56, 1202},  
03668 {"Lt Weathered Blue", 0xFFAEB8C3, 1203},  
03669 {"Pastel Jade", 0xFFA8C8BC, 1204},  
03670 {"Med Jade", 0xFF6E90A5, 1205},  
03671 {"Dark Jade", 0xFF1E6E6F, 1206},  
03672 {"Sea Foam Green", 0xFF80A388, 1207},  
03673 {"Mallard Green", 0xFF0C3D03, 1208},  
03674 {"Lt Avocado", 0xFFBDD163, 1209},  
03675 {"Dk Army Green", 0xFF273B00, 1210},  
03676 {"Lt Khaki", 0xFF95A490, 1211},  
03677 {"Khaki", 0xFF63632D, 1212},  
03678 {"Taupe", 0xFFB9A096, 1213},  
03679 {"Med Chestnut", 0xFF642828, 1214},  
03680 {"Blackberry", 0xFF500A1E, 1215},  
03681 {"Med Maple", 0xFFAC1C01, 1216},  
03682 {"Chestnut", 0xFF971F01, 1217},  
03683 {"Silver Gray", 0xFFDFDFCB, 1218},  
03684 {"Gray", 0xFF98888C, 1219},  
03685 {"Charcoal Gray", 0xFF765960, 1220},  
03686 {"Lt Baby Blue", 0xFFD1DBFF, 1222},
```

```
03687 {"Baby Blue Tint", 0xFFDCE0F1, 1223},  
03688 {"Bright Pink", 0xFFFF0A0B9, 1224},  
03689 {"Pastel Pink", 0xFFFFACBCB, 1225},  
03690 {"Dkl Periwinkle", 0xFF57369E, 1226},  
03691 {"Gold Green", 0xFFAF8901, 1227},  
03692 {"Drab Green", 0xFF96AA8B, 1228},  
03693 {"Lt Putty", 0xFFE0DBDB, 1229},  
03694 {"Dk Teal", 0xFF0B4133, 1230},  
03695 {"Med Rose", 0xFFE5326A, 1231},  
03696 {"Classic Green", 0xFF193207, 1232},  
03697 {"Ocean Teal", 0xFF0D2210, 1233},  
03698 {"Almost Black", 0xFF3C1B1F, 1234},  
03699 {"Deep Purple", 0xFF783298, 1235},  
03700 {"Lt Silver", 0xFFEAEB4E4, 1236},  
03701 {"Deep Mauve", 0xFFBC3D2C, 1237},  
03702 {"Orange Sunrise", 0xFFFF8300, 1238},  
03703 {"Apricot", 0xFFFFAB57, 1239},  
03704 {"Smokey Grey", 0xFF74586C, 1240},  
03705 {"Nassau Blue", 0xFF543A8D, 1242},  
03706 {"Orange Flame", 0xFFFF0000, 1246},  
03707 {"Mahogany", 0xFF660000, 1247},  
03708 {"Med Pastel Blue", 0xFFD2E6F0, 1248},  
03709 {"Cornflower Blue", 0xFF62AADC, 1249},  
03710 {"Duck Wing Blue", 0xFF275C70, 1250},  
03711 {"Bright Turquoise", 0xFFF306F75, 1251},  
03712 {"Bright Peacock", 0xFF09A1A8, 1252},  
03713 {"Dk Sapphire", 0xFF1B4CA4, 1253},  
03714 {"Dusty Lavender", 0xFFE6B9F5, 1254},  
03715 {"Deep Orchid", 0xFFBE1982, 1255},  
03716 {"Sweet Pink", 0xFFEB8296, 1256},  
03717 {"Deep Coral", 0xFFE60041, 1257},  
03718 {"Coral Reed", 0xFFF0C4A0, 1258},  
03719 {"Salmon Peach", 0FFE28264, 1259},  
03720 {"Red Jubilee", 0xFFB30000, 1263},  
03721 {"Cognac", 0xFF6A0000, 1264},  
03722 {"Burnt Toast", 0xFF9B6B2C, 1265},  
03723 {"Toast", 0xFF9C6D45, 1266},  
03724 {"Mink Brown", 0xFF864C31, 1267},  
03725 {"Light Gray Khaki", 0xFFFFEF0E, 1268},  
03726 {"Dk Gray Khaki", 0xFFB7B7AF, 1270},  
03727 {"Evergreen", 0xFF3C4F31, 1271},  
03728 {"Hedge Green", 0xFF4A4A19, 1272},  
03729 {"Nile Green", 0xFF5C9A1A, 1274},  
03730 {"Sea Mist", 0xFFE0E6C8, 1275},  
03731 {"Pistachio", 0xFF70770F, 1276},  
03732 {"Ivy Green", 0xFF027602, 1277},  
03733 {"Bright Green", 0xFF00AF38, 1278},  
03734 {"Willow Green", 0xFF93D16C, 1279},  
03735 {"Dk Willow Green", 0XF46B774, 1280},  
03736 {"Slate Gray", 0xFF483D59, 1283},  
03737 {"Dk Winter Sky", 0xFF466E78, 1284},  
03738 {"Dk Sage Green", 0xFF134F45, 1285},  
03739 {"Dk French Green", 0xFF343213, 1286},  
03740 {"French Green", 0xFF415545, 1287},  
03741 {"Aqua", 0xFF0FA56F, 1288},  
03742 {"Ice Blue", 0xFFDCEBF0, 1289},  
03743 {"Winter Sky", 0xFF727483, 1291},  
03744 {"Heron Blue", 0xFFD1DCFA, 1292},  
03745 {"Deep Nassau Blue", 0xFFF44235D, 1293},  
03746 {"Deep Slate Gray", 0xFF412044, 1294},  
03747 {"Sterling", 0xFF82878C, 1295},  
03748 {"Hyacinth", 0xFFD2AAF0, 1296},  
03749 {"Lt Plum", 0xFF735A64, 1297},  
03750 {"Dk Plum", 0xFF644664, 1298},  
03751 {"Purple Shadow", 0xFF411446, 1299},  
03752 {"Plum", 0xFF7E1E46, 1300},  
03753 {"Deep Eggplant", 0xFF320046, 1301},  
03754 {"Eggplant", 0xFF6E0A96, 1302},  
03755 {"Dewberry", 0xFFB47364, 1304},  
03756 {"Sage Green", 0xFFAEC6BB, 1305},  
03757 {"Gun Metal Gray", 0xFF7E6C7C, 1306},  
03758 {"Petal Pink", 0xFFDB6478, 1307},  
03759 {"Magenta", 0xFF782346, 1309},  
03760 {"Mulberry", 0xFF961A32, 1311},  
03761 {"Wine", 0xFF840000, 1312},  
03762 {"Bittersweet", 0xFFFFC8F0C, 1313},  
03763 {"Poppy", 0xFFFF0000, 1317}, /* TODO: duplicate case value */  
03764 {"Gray Khaki", 0xFFCBBCBD, 1321},  
03765 {"Chartreuse", 0xFF818901, 1322},  
03766 {"Whisper Gray", 0xFFF8F5F1, 1325},  
03767 {"Dk Whisper Gray", 0xFFD5C7C3, 1327},  
03768 {"Nickel Gray", 0xFFC0B2B7, 1328},  
03769 {"Dk Nickel Gray", 0xFFABA0A8, 1329},  
03770 {"Pale Green", 0xFFEDF6D4, 1331},  
03771 {"Deep Chartreuse", 0xFF868105, 1332},  
03772 {"Sunflower Gold", 0xFFFB600, 1333},  
03773 {"Green Peacock", 0xFF349669, 1503},
```

```

03774 {"Putty", 0xFFC1CBB9, 1508},
03775 {"Lime Green", 0xFF7AB31D, 1510},
03776 {"Deep Rose", 0xFFEE5078, 1511},
03777 {"Wild Peacock", 0xFF007A67, 1513},
03778 {"Rosebud", 0xFFFF8CCB, 1515},
03779 {"Coachman Green", 0xFF014F3A, 1517},
03780 {"Lt Rose", 0xFFCD054D, 1533},
03781 {"Sapphire", 0xFF347DCB, 1534},
03782 {"Team Blue", 0xFF23238B, 1535},
03783 {"Midnight Teal", 0xFF081705, 1536},
03784 {"Peach Fluff", 0xFFFFD6C7, 1543},
03785 {"Purple Accent", 0xFF9C6484, 1545},
03786 {"Flax", 0xFFE6AE6F, 1549},
03787 {"Desert Cactus", 0xFF6C8E87, 1550},
03788 {"Ocean Aqua", 0xFF80B0AE, 1551},
03789 {"Dk Desert Cactus", 0xFF6C7C71, 1552},
03790 {"Purple Passion", 0xFF8C748C, 1554},
03791 {"Tea Rose", 0xFFEB7183, 1558},
03792 {"Marine Aqua", 0xFF68E0F8, 1560},
03793 {"Deep Hyacinth", 0xFFB58CC7, 1561},
03794 {"Shrimp", 0xFFFFAD2AA, 1800},
03795 {"Flesh", 0xFFFFADC96, 1801},
03796 {"Soft Blush", 0xFFFFC896, 1802},
03797 {"Island Peach", 0xFFFF9B6E, 1803},
03798 {"Bayou Blue", 0xFF375A73, 1804},
03799 {"Ocean View", 0xFF28505A, 1805},
03800 {"Madras Blue", 0xFFA0B9C3, 1806},
03801 {"Soft Heather", 0xFFB49682, 1807},
03802 {"Velvet Slipper", 0xFFD2AF9B, 1808},
03803 {"Iced Mauve", 0xFFA07D82, 1809},
03804 {"Wild Mulberry", 0xFF645055, 1810},
03805 {"Wineberry", 0xFF3C2837, 1811},
03806 {"Wildflower", 0xFF6E2D5A, 1812},
03807 {"Plum Wine", 0xFF6E2D41, 1813},
03808 {"Orchid Kiss", 0xFFAF4B69, 1814},
03809 {"Japanese Fern", 0xFF91B432, 1815},
03810 {"Honeydew", 0xFFD7F58C, 1816},
03811 {"Lemon Grass", 0xFFAAAF14, 1817},
03812 {"Fairway Mist", 0xFFC8F58C, 1818},
03813 {"Outback", 0xFFC3913C, 1819},
03814 {"Fruit Shake", 0xFFC38C73, 1820},
03815 {"Creamy Peach", 0xFFFFAC896, 1821},
03816 {"Toffee", 0xFFF965A37, 1822},
03817 {"Cocoa", 0xFF965A28, 1823},
03818 {"Gentle Rain", 0xFFD2C3AF, 1824},
03819 {"Barnyard Grass", 0xFF5F9619, 1825},
03820 {"Galley Gold", 0xFFAA820A, 1826},
03821 {"Coral Sunset", 0xFFFFF643C, 1827},
03822 {"Seashell", 0xFFFFE6AA, 1828},
03823 {"Crème Brulee", 0xFFFF0EBA5, 1829},
03824 {"Lilac", 0xFFB47396, 1830},
03825 {"Liimeade", 0xFF91E12D, 1831},
03826 {"Desert Glow", 0xFFE19119, 1832},
03827 {"Pumpkin Pie", 0xFFD25F00, 1833},
03828 {"Pea Soup", 0xFFAFAA05, 1834},
03829 {"Peapod Green", 0xFF6E8205, 1835},
03830 {"Loden Green", 0xFF3C4B05, 1836},
03831 {"Lt. Cocoa", 0xFF9B735A, 1837},
03832 {"Cocoa Cream", 0xFFCDA47D, 1838},
03833 {"Cameo", 0xFF87C887, 1839},
03834 {"Sand", 0xFF9E6CA, 508},
03835 {"Bone", 0xFFDFD3DA, 520},
03836 {"Dark Ash", 0xFF5D3446, 1241},
03837 {"Spring Moss", 0xFFEOC63B, 1243},
03838 {"Summer Gold", 0xFFDDAB00, 1260},
03839 {"Dk. Autumn Gold", 0xFA98803, 1262},
03840 {"Mushroom", 0xFFAC8783, 1269},
03841 {"Dark Forest", 0xFF36361F, 1273},
03842 {"Watermelon", 0xFFFFA5F7F, 1303},
03843 {"Caribbean Mist", 0xFFA3C2D7, 1644},
03844 {"END", 0, -1}
03845 };
03846
03847 thread_color ThreadArt_Rayon_codes[] = {
03848 {"END", 0, -1}
03849 };
03850
03851 thread_color ThreadArt_Polyester_codes[] = {
03852 {"END", 0, -1}
03853 };
03854
03855 thread_color ThreaDelight_Polyester_codes[] = {
03856 {"END", 0, -1}
03857 };
03858
03859 thread_color Z102_Isacord_Polyester_codes[] = {
03860 {"?", 0xFFFF8FFFF, 17},

```

```
03861 {"?", 0xFF000000, 20},  
03862 {"?", 0xFFB7BAB0, 105},  
03863 {"?", 0xFF73787A, 108},  
03864 {"?", 0xFF454B58, 138},  
03865 {"?", 0xFF9EA9A6, 142},  
03866 {"?", 0xFFC8C6BD, 150},  
03867 {"?", 0xFFFFAEE5C, 220},  
03868 {"?", 0xFFE5CB4F, 221},  
03869 {"?", 0xFFFFF46A, 230},  
03870 {"?", 0xFFFFF9D9, 270},  
03871 {"?", 0xFFFFDC00, 311},  
03872 {"?", 0xFF624F00, 345},  
03873 {"?", 0xFFB8B25A, 352},  
03874 {"?", 0xFF8D8F5B, 453},  
03875 {"?", 0xFFFFF4A5, 520},  
03876 {"?", 0xFFB98E03, 542},  
03877 {"?", 0FFE4C180, 651},  
03878 {"?", 0xFFC5BFA6, 672},  
03879 {"?", 0xFF96836D, 722},  
03880 {"?", 0xFF4E3500, 747},  
03881 {"?", 0xFFDDCBA5, 761},  
03882 {"?", 0xFF605840, 776},  
03883 {"?", 0xFFFFFAF02, 800},  
03884 {"?", 0xFFFF6AE32, 811},  
03885 {"?", 0xFFC89334, 822},  
03886 {"?", 0xFFE59300, 824},  
03887 {"?", 0xFFC89340, 832},  
03888 {"?", 0xFF9E947F, 873},  
03889 {"?", 0FFC8700B, 922},  
03890 {"?", 0xFFB5704, 931},  
03891 {"?", 0FFE19072, 1061},  
03892 {"?", 0FFF8101, 1102},  
03893 {"?", 0FFB1500A, 1115},  
03894 {"?", 0FFC09C72, 1123},  
03895 {"?", 0FF843D07, 1134},  
03896 {"?", 0FFD8A67D, 1141},  
03897 {"?", 0FFB2421B, 1154},  
03898 {"?", 0FFF7319, 1300},  
03899 {"?", 0FFF3D1B, 1305},  
03900 {"?", 0FFBA4005, 1311},  
03901 {"?", 0FFC73C13, 1312},  
03902 {"?", 0FFE66B21, 1332},  
03903 {"?", 0FF3D1C11, 1346},  
03904 {"?", 0FFFBC95, 1351},  
03905 {"?", 0FFFCC93, 1362},  
03906 {"?", 0FF373732, 1375},  
03907 {"?", 0FFFFAF94, 1532},  
03908 {"?", 0FF5B4032, 1565},  
03909 {"?", 0FFF6046, 1600},  
03910 {"?", 0FFF6D71, 1753},  
03911 {"?", 0FFEBAAE, 1755},  
03912 {"?", 0FFEB2D2B, 1805},  
03913 {"?", 0FFF988F, 1840},  
03914 {"?", 0FF434331, 1874},  
03915 {"?", 0FFC11914, 1902},  
03916 {"?", 0FFC8100D, 1903},  
03917 {"?", 0FFBF0A21, 1906},  
03918 {"?", 0FFD23C3E, 1921},  
03919 {"?", 0FF8F8C93, 1972},  
03920 {"?", 0FFA31A1C, 2011},  
03921 {"?", 0FF4D0308, 2115},  
03922 {"?", 0FFFCDCC, 2155},  
03923 {"?", 0FF871C45, 2500},  
03924 {"?", 0FFDB4083, 2508},  
03925 {"?", 0FFF668A, 2520},  
03926 {"?", 0FFC91243, 2521},  
03927 {"?", 0FFF0B6, 2530},  
03928 {"?", 0FFFA5B9, 2550},  
03929 {"?", 0FF626C7E, 2674},  
03930 {"?", 0FF5E1942, 2711},  
03931 {"?", 0FF33001D, 2715},  
03932 {"?", 0FFA57B8D, 2764},  
03933 {"?", 0FF702A69, 2810},  
03934 {"?", 0FFB385BC, 2830},  
03935 {"?", 0FF240047, 2900},  
03936 {"?", 0FF724593, 2910},  
03937 {"?", 0FF634D86, 2920},  
03938 {"?", 0FF000136, 3110},  
03939 {"?", 0FF000021, 3355},  
03940 {"?", 0FF054ABD, 3522},  
03941 {"?", 0FF1C005D, 3541},  
03942 {"?", 0FF001F71, 3544},  
03943 {"?", 0FF002E5E, 3622},  
03944 {"?", 0FF71AAD8, 3630},  
03945 {"?", 0FF001748, 3644},  
03946 {"?", 0FFC8DBE4, 3650},  
03947 {"?", 0FF9FC7DF, 3730},
```

```

03948 {"?", 0xFF082E4D, 3743},
03949 {"?", 0xFF98B0BC, 3750},
03950 {"?", 0xFF23679C, 3810},
03951 {"?", 0xFF3D657B, 3842},
03952 {"?", 0xFF00669F, 3901},
03953 {"?", 0xFF47AEDD, 3910},
03954 {"?", 0xFFBBDFFB, 3962},
03955 {"?", 0xFFBABEB7, 3971},
03956 {"?", 0xFF015D7E, 4032},
03957 {"?", 0xFFD5DDDB, 4071},
03958 {"?", 0xFF888D8E, 4073},
03959 {"?", 0xFF007CA6, 4103},
03960 {"?", 0xFF3EBBC8, 4111},
03961 {"?", 0xFF005C79, 4116},
03962 {"?", 0xFF80CCD8, 4240},
03963 {"?", 0xFFACCECT, 4250},
03964 {"?", 0xFF006E74, 4410},
03965 {"?", 0xFF002A29, 4515},
03966 {"?", 0xFF38A4AE, 4620},
03967 {"?", 0xFFAFD8CD, 5050},
03968 {"?", 0xFF149B7B, 5210},
03969 {"?", 0xFF7AC8AF, 5220},
03970 {"?", 0xFF187166, 5233},
03971 {"?", 0xFF004B23, 5374},
03972 {"?", 0xFF006835, 5415},
03973 {"?", 0xFF5C9C51, 5531},
03974 {"?", 0xFF0E9543, 5613},
03975 {"?", 0xFF5E7A17, 5833},
03976 {"?", 0xFF225926, 5944},
03977 {"?", 0xFFBCD633, 6011},
03978 {"?", 0xFFBBCD91, 6051},
03979 {"?", 0xFF978B3C, 6133},
03980 {"END", 0, -1}
03981 };
03982
03983 thread_color *brand_codes[] = {
03984     Arc_Polyester_codes,          /* 0 */
03985     Arc_Rayon_codes,             /* 1 */
03986     CoatsAndClark_Rayon_codes,   /* 2 */
03987     Exquisite_Polyester_codes,   /* 3 */
03988     Fufu_Polyester_codes,        /* 4 */
03989     Fufu_Rayon_codes,            /* 5 */
03990     Hemingworth_Polyester_codes, /* 6 */
03991     Isacord_Polyester_codes,    /* 7 */
03992     Isafil_Rayon_codes,          /* 8 */
03993     Marathon_Polyester_codes,   /* 9 */
03994     Marathon_Rayon_codes,       /* 10 */
03995     Madeira_Polyester_codes,    /* 11 */
03996     Madeira_Rayon_codes,         /* 12 */
03997     Metro_Polyester_codes,      /* 13 */
03998     Pantone_codes,              /* 14 */
03999     RobisonAnton_Polyester_codes, /* 15 */
04000     RobisonAnton_Rayon_codes,   /* 16 */
04001     Sigma_Polyester_codes,      /* 17 */
04002     Sulky_Rayon_codes,           /* 18 */
04003     ThreadArt_Rayon_codes,      /* 19 */
04004     ThreadArt_Polyester_codes,   /* 20 */
04005     ThreadDelight_Polyester_codes, /* 21 */
04006     Z102_Isacord_Polyester_codes, /* 22 */
04007     svg_color_codes             /* 23 */
04008 },
04009 #endif
04010
04011
04012 int threadColor(const char *name, int brand)
04013 {
04014     int i;
04015     for (i = 0; brand_codes[brand][i].manufacturer_code >= 0; i++) {
04016         if (!strcmp(brand_codes[brand][i].name, name)) {
04017             return brand_codes[brand][i].hex_code;
04018         }
04019     }
04020     return -1;
04021 }
04022
04023 int threadColorNum(unsigned int color, int brand)
04024 {
04025     int i;
04026     for (i = 0; brand_codes[brand][i].manufacturer_code >= 0; i++) {
04027         if (brand_codes[brand][i].hex_code == color) {
04028             return brand_codes[brand][i].manufacturer_code;
04029         }
04030     }
04031
04032     return -1;
04033 }
04034

```

```
04035 const char* threadColorName(unsigned int color, int brand)
04036 {
04037     int i;
04038     for (i = 0; brand_codes[brand][i].manufacturer_code >= 0; i++) {
04039         if (brand_codes[brand][i].hex_code == color) {
04040             return brand_codes[brand][i].name;
04041         }
04042     }
04043
04044     return "COLOR NOT FOUND";
04045 }
```


Index

_bcf_directory, 5
dirEntries, 5
maxNumberOfDirectoryEntries, 5
_bcf_directory_entry, 6
childId, 6
CLSID, 6
colorFlag, 6
creationTime, 7
directoryEntryName, 7
directoryEntryNameLength, 7
leftSiblingId, 7
modifiedTime, 7
next, 7
objectType, 8
rightSiblingId, 8
startingSectorLocation, 8
stateBits, 8
streamSize, 8
streamSizeHigh, 8
_bcf_file, 9
difat, 9
directory, 9
fat, 9
header, 10
_bcf_file_difat, 10
fatSectorCount, 10
fatSectorEntries, 10
sectorSize, 11
_bcf_file_fat, 11
fatEntries, 11
fatEntryCount, 11
numberOfEntriesInFatSector, 12
_bcf_file_header, 12
byteOrder, 13
CLSID, 13
firstDifatSectorLocation, 13
firstDirectorySectorLocation, 13
firstMiniFATSectorLocation, 13
majorVersion, 13
miniSectorShift, 14
miniStreamCutoffSize, 14
minorVersion, 14
numberOfDifatSectors, 14
numberOfDirectorySectors, 14
numberOfFATSectors, 14
numberOfMiniFatSectors, 15
reserved1, 15
reserved2, 15
sectorShift, 15
signature, 15
transactionSignatureNumber, 15
_dxfColorTable
embroidery.h, 145
thread-color.c, 340
_vp3Hoop, 16
bottom, 16
bottom2, 17
byte1, 17
byte2, 17
byte3, 17
height, 17
left, 17
left2, 18
numberOfBytesRemaining, 18
numberOfColors, 18
right, 18
right2, 18
threadLength, 18
top, 19
top2, 19
unknown2, 19
unknown3, 19
unknown4, 19
width, 19
xOffset, 20
yOffset, 20
alphabet
LSYSTEM, 60
arc
EmbGeometry_, 34
Arc_Polyester
embroidery.h, 92
Arc_Rayon
embroidery.h, 92
array.c
embArray_addArc, 71
embArray_addCircle, 72
embArray_addEllipse, 72
embArray_addFlag, 72
embArray_addLine, 72
embArray_addPath, 72
embArray_addPoint, 73
embArray_addPolygon, 73
embArray_addPolyline, 73
embArray_addRect, 73
embArray_addStitch, 73
embArray_addVector, 74
embArray_copy, 74

embArray_create, 74
 embArray_free, 74
 embArray_resize, 74
attributeOffset
 VipHeader_, 68
auxFormat
 ThredExtension_, 65
axiom
 LSYSTEM, 60
b
 EmbColor_, 30
bcf_difat_create
 embroidery_internal.h, 188
 main.c, 285
bcf_directory
 embroidery_internal.h, 185
bcf_directory_entry
 embroidery_internal.h, 185
bcf_directory_free
 embroidery_internal.h, 188
 main.c, 285
bcf_file
 embroidery_internal.h, 186
bcf_file_difat
 embroidery_internal.h, 186
bcf_file_difat_free
 embroidery_internal.h, 188
bcf_file_fat
 embroidery_internal.h, 186
bcf_file_fat_free
 embroidery_internal.h, 189
bcf_file_free
 embroidery_internal.h, 189
 main.c, 285
bcf_file_header
 embroidery_internal.h, 186
bcfFile_read
 embroidery_internal.h, 189
 main.c, 285
bcfFileFat_create
 embroidery_internal.h, 189
 main.c, 285
bcfFileHeader_isValid
 embroidery_internal.h, 189
bcfFileHeader_read
 embroidery_internal.h, 189
 main.c, 286
beziers
 EmbSpline_, 51
binaryReadString
 embroidery_internal.h, 190
 main.c, 286
binaryReadUnicodeString
 embroidery_internal.h, 190
 main.c, 286
binaryWriteInt
 formats.c, 257
binaryWriteIntBE
 formats.c, 257
binaryWriteShort
 formats.c, 258
binaryWriteUInt
 embroidery_internal.h, 190
 formats.c, 258
binaryWriteUIntBE
 formats.c, 258
binaryWriteUShort
 formats.c, 258
binaryWriteUShortBE
 formats.c, 258
bit_position
 Compress, 21
bits_total
 Compress, 21
black_thread
 embroidery.h, 145
 main.c, 293
block_elements
 Compress, 21
bottom
 _vp3Hoop, 16
 EmbRect_, 49
bottom2
 _vp3Hoop, 17
brand_codes
 thread-color.c, 340
brand_codes_files
 thread-color.c, 340
BULGETOCONTROL
 embroidery_internal.h, 162
BULGETOEND
 embroidery_internal.h, 163
byte1
 _vp3Hoop, 17
byte2
 _vp3Hoop, 17
byte3
 _vp3Hoop, 17
byteOrder
 _bcf_file_header, 13
catalogNumber
 EmbThread_, 55
center
 EmbCircle_, 29
 EmbEllipse_, 31
character_huffman
 Compress, 21
character_length_huffman
 Compress, 21
check_for_color_file
 EmbFormatList_, 32
check_header_present
 embroidery_internal.h, 190
 main.c, 286
childId
 _bcf_directory_entry, 6

CHUNK_SIZE
embroidery.h, 93

circle
EmbGeometry_, 34

CLSID
_bcf_directory_entry, 6
_bcf_file_header, 13

CoatsAndClark_Rayon
embroidery.h, 93

color
EmbGeometry_, 34
EmbLine_, 41
EmbPath_, 44
EmbPoint_, 47
EmbStitch_, 52
EmbThread_, 55

color_only
EmbFormatList_, 32

colorCode
StxThread_, 61
SubDescriptor_, 62

colorFlag
_bcf_directory_entry, 6

colorLength
VipHeader_, 69

colorName
StxThread_, 61
SubDescriptor_, 62

CompoundFileDirectory
embroidery_internal.h, 190
main.c, 286

CompoundFileDirectoryEntry
embroidery_internal.h, 191
main.c, 287

CompoundFileSector_DIFAT_Sector
embroidery_internal.h, 163

CompoundFileSector_EndOfChain
embroidery_internal.h, 163

CompoundFileSector_FAT_Sector
embroidery_internal.h, 163

CompoundFileSector_FreeSector
embroidery_internal.h, 163

CompoundFileSector_MaxRegSector
embroidery_internal.h, 163

CompoundFileStreamId_MaxRegularStreamId
embroidery_internal.h, 164

CompoundFileStreamId_NoStream
embroidery_internal.h, 164

Compress, 20
bit_position, 21
bits_total, 21
block_elements, 21
character_huffman, 21
character_length_huffman, 21
distance_huffman, 21
input_data, 22
input_length, 22

compress
embroidery_internal.h, 186

compress.c
compress_get_bits, 79
compress_get_position, 79
compress_get_token, 79
compress_init, 79
compress_load_block, 79
compress_load_character_huffman, 79
compress_load_character_length_huffman, 80
compress_load_distance_huffman, 80
compress_peek, 80
compress_pop, 80
compress_read_variable_length, 80
huffman_build_table, 81
huffman_lookup, 81
huffman_lookup_data, 82
hus_compress, 81
hus_decompress, 81

compress_get_bits
compress.c, 79
embroidery_internal.h, 191

compress_get_position
compress.c, 79
embroidery_internal.h, 191

compress_get_token
compress.c, 79
embroidery_internal.h, 191

compress_init
compress.c, 79

compress_load_block
compress.c, 79
embroidery_internal.h, 191

compress_load_character_huffman
compress.c, 79
embroidery_internal.h, 192

compress_load_character_length_huffman
compress.c, 80
embroidery_internal.h, 192

compress_load_distance_huffman
compress.c, 80
embroidery_internal.h, 192

compress_peek
compress.c, 80

compress_pop
compress.c, 80
embroidery_internal.h, 192

compress_read_variable_length
compress.c, 80
embroidery_internal.h, 192

constants
LSYSTEM, 60

control1
EmbBezier_, 27

control2
EmbBezier_, 27

convert
embroidery.h, 120
pattern.c, 316

copy_trim
 embroidery_internal.h, 193
 main.c, 287
 count
 EmbArray_, 25
 create_test_file_1
 embroidery_internal.h, 193
 create_test_file_2
 embroidery_internal.h, 193
 create_test_file_3
 embroidery_internal.h, 193
 creationTime
 _bcf_directory_entry, 7
 creatorName
 ThredExtension_, 65
 CSV_EXPECT
 embroidery_internal.h, 187
 CSV_EXPECT_COMMA
 embroidery_internal.h, 188
 CSV_EXPECT_NULL
 embroidery_internal.h, 188
 CSV_EXPECT_QUOTE1
 embroidery_internal.h, 188
 CSV_EXPECT_QUOTE2
 embroidery_internal.h, 188
 CSV_MODE
 embroidery_internal.h, 188
 CSV_MODE_COMMENT
 embroidery_internal.h, 188
 CSV_MODE_NULL
 embroidery_internal.h, 188
 CSV_MODE_STITCH
 embroidery_internal.h, 188
 CSV_MODE_THREAD
 embroidery_internal.h, 188
 CSV_MODE_VARIABLE
 embroidery_internal.h, 188
 CUBICTOCONTROL1
 embroidery_internal.h, 164
 CUBICTOCONTROL2
 embroidery_internal.h, 164
 CUBICTOEND
 embroidery_internal.h, 164
 currentColorIndex
 EmbPattern_, 45

 data
 EmblImage_, 38
 day
 EmbTime_, 56
 decode_t01_record
 embroidery_internal.h, 193
 encoding.c, 232
 decode_tajima_ternary
 embroidery_internal.h, 193
 encoding.c, 232
 decodeNewStitch
 embroidery_internal.h, 194
 encoding.c, 232

 default_value
 Huffman, 58
 degrees
 embroidery.h, 120
 description
 EmbFormatList_, 32
 EmbThread_, 55
 difat
 _bcf_file, 9
 difatEntriesInHeader
 main.c, 293
 dimensions
 EmblImage_, 38
 directory
 _bcf_file, 9
 directoryEntryName
 _bcf_directory_entry, 7
 directoryEntryNameLength
 _bcf_directory_entry, 7
 dirEntries
 _bcf_directory, 5
 distance_huffman
 Compress, 21
 dragon_curve
 fill.c, 240
 dstJumpsPerTrim
 EmbPattern_, 45
 dxf_color
 embroidery.h, 93
 DXF_VERSION_2000
 embroidery_internal.h, 165
 DXF_VERSION_2002
 embroidery_internal.h, 165
 DXF_VERSION_2004
 embroidery_internal.h, 165
 DXF_VERSION_2006
 embroidery_internal.h, 165
 DXF_VERSION_2007
 embroidery_internal.h, 165
 DXF_VERSION_2009
 embroidery_internal.h, 165
 DXF_VERSION_2010
 embroidery_internal.h, 166
 DXF_VERSION_2013
 embroidery_internal.h, 166
 DXF_VERSION_R10
 embroidery_internal.h, 166
 DXF_VERSION_R11
 embroidery_internal.h, 166
 DXF_VERSION_R12
 embroidery_internal.h, 166
 DXF_VERSION_R13
 embroidery_internal.h, 166
 DXF_VERSION_R14
 embroidery_internal.h, 167
 DXF_VERSION_R15
 embroidery_internal.h, 167
 DXF_VERSION_R18

embroidery_internal.h, 167
DXF_VERSION_R21
embroidery_internal.h, 167
DXF_VERSION_R24
embroidery_internal.h, 167
DXF_VERSION_R27
embroidery_internal.h, 167

ELEMENT_A
embroidery_internal.h, 168
ELEMENT_ANIMATE
embroidery_internal.h, 168
ELEMENT_ANIMATECOLOR
embroidery_internal.h, 168
ELEMENT_ANIMATEMOTION
embroidery_internal.h, 168
ELEMENT_ANIMATETRANSFORM
embroidery_internal.h, 168
ELEMENT_ANIMATION
embroidery_internal.h, 168
ELEMENT_AUDIO
embroidery_internal.h, 169
ELEMENT_CIRCLE
embroidery_internal.h, 169
ELEMENT_DEFS
embroidery_internal.h, 169
ELEMENT_DESC
embroidery_internal.h, 169
ELEMENT_DISCARD
embroidery_internal.h, 169
ELEMENT_ELLIPSE
embroidery_internal.h, 169
ELEMENT_FONT
embroidery_internal.h, 170
ELEMENT_FONT_FACE
embroidery_internal.h, 170
ELEMENT_FONT_FACE_SRC
embroidery_internal.h, 170
ELEMENT_FONT_FACE_URI
embroidery_internal.h, 170
ELEMENT_FOREIGN_OBJECT
embroidery_internal.h, 170
ELEMENT_G
embroidery_internal.h, 170
ELEMENT_GLYPH
embroidery_internal.h, 171
ELEMENT_HANDLER
embroidery_internal.h, 171
ELEMENT_HKERN
embroidery_internal.h, 171
ELEMENT_IMAGE
embroidery_internal.h, 171
ELEMENT_LINE
embroidery_internal.h, 171
ELEMENT_LINEAR_GRADIENT
embroidery_internal.h, 171
ELEMENT_LISTENER
embroidery_internal.h, 172
ELEMENT_METADATA
embroidery_internal.h, 172
ELEMENT_MISSING_GLYPH
embroidery_internal.h, 172
ELEMENT_MPATH
embroidery_internal.h, 172
ELEMENT_PATH
embroidery_internal.h, 172
ELEMENT_POLYGON
embroidery_internal.h, 172
ELEMENT_POLYLINE
embroidery_internal.h, 173
ELEMENT_PREFETCH
embroidery_internal.h, 173
ELEMENT_RADIAL_GRADIENT
embroidery_internal.h, 173
ELEMENT_RECT
embroidery_internal.h, 173
ELEMENT_SCRIPT
embroidery_internal.h, 173
ELEMENT_SET
embroidery_internal.h, 173
ELEMENT_SOLID_COLOR
embroidery_internal.h, 174
ELEMENT_STOP
embroidery_internal.h, 174
ELEMENT_SVG
embroidery_internal.h, 174
ELEMENT_SWITCH
embroidery_internal.h, 174
ELEMENT_TBREAK
embroidery_internal.h, 174
ELEMENT_TEXT
embroidery_internal.h, 174
ELEMENT_TEXT_AREA
embroidery_internal.h, 175
ELEMENT_TITLE
embroidery_internal.h, 175
ELEMENT_TSPAN
embroidery_internal.h, 175
ELEMENT_USE
embroidery_internal.h, 175
ELEMENT_VIDEO
embroidery_internal.h, 175
ELEMENT_XML
embroidery_internal.h, 175
ellipse
EmbGeometry_, 35
ELLIPSETOEND
embroidery_internal.h, 176
ELLIPSETORAD
embroidery_internal.h, 176
EMB_ARC
embroidery.h, 93
EMB_ARRAY
embroidery.h, 93
EMB_BIG_ENDIAN
embroidery_internal.h, 176
EMB_CIRCLE

embroidery.h, 93
EMB_DIM_DIAMETER
 embroidery.h, 94
EMB_DIM_LEADER
 embroidery.h, 94
EMB_ELLIPSE
 embroidery.h, 94
emb_error
 embroidery.h, 145
 main.c, 294
EMB_FLAG
 embroidery.h, 94
EMB_FORMAT_100
 embroidery.h, 94
EMB_FORMAT_10O
 embroidery.h, 94
EMB_FORMAT_ART
 embroidery.h, 95
EMB_FORMAT_BMC
 embroidery.h, 95
EMB_FORMAT_BRO
 embroidery.h, 95
EMB_FORMAT_CND
 embroidery.h, 95
EMB_FORMAT_COL
 embroidery.h, 95
EMB_FORMAT_CSD
 embroidery.h, 95
EMB_FORMAT_CSV
 embroidery.h, 96
EMB_FORMAT_DAT
 embroidery.h, 96
EMB_FORMAT_DEM
 embroidery.h, 96
EMB_FORMAT_DSB
 embroidery.h, 96
EMB_FORMAT_DST
 embroidery.h, 96
EMB_FORMAT_DSZ
 embroidery.h, 96
EMB_FORMAT_DXF
 embroidery.h, 97
EMB_FORMAT_EDR
 embroidery.h, 97
EMB_FORMAT_EMD
 embroidery.h, 97
EMB_FORMAT_EXP
 embroidery.h, 97
EMB_FORMAT_EXY
 embroidery.h, 97
EMB_FORMAT_EYS
 embroidery.h, 97
EMB_FORMAT_FXY
 embroidery.h, 98
EMB_FORMAT_GC
 embroidery.h, 98
EMB_FORMAT_GNC
 embroidery.h, 98
EMB_FORMAT_GT
 embroidery.h, 98
EMB_FORMAT_HUS
 embroidery.h, 98
EMB_FORMAT_INB
 embroidery.h, 98
EMB_FORMAT_INF
 embroidery.h, 99
EMB_FORMAT_JEF
 embroidery.h, 99
EMB_FORMAT_KSM
 embroidery.h, 99
EMB_FORMAT_MAX
 embroidery.h, 99
EMB_FORMAT_MIT
 embroidery.h, 99
EMB_FORMAT_NEW
 embroidery.h, 99
EMB_FORMAT_OFM
 embroidery.h, 100
EMB_FORMAT_PCD
 embroidery.h, 100
EMB_FORMAT_PCM
 embroidery.h, 100
EMB_FORMAT_PCQ
 embroidery.h, 100
EMB_FORMAT_PCS
 embroidery.h, 100
EMB_FORMAT_PEC
 embroidery.h, 100
EMB_FORMAT_PEL
 embroidery.h, 101
EMB_FORMAT_PEM
 embroidery.h, 101
EMB_FORMAT_PES
 embroidery.h, 101
EMB_FORMAT_PHB
 embroidery.h, 101
EMB_FORMAT_PHC
 embroidery.h, 101
EMB_FORMAT_PLT
 embroidery.h, 101
EMB_FORMAT_RGB
 embroidery.h, 102
EMB_FORMAT_SEW
 embroidery.h, 102
EMB_FORMAT_SHV
 embroidery.h, 102
EMB_FORMAT_SST
 embroidery.h, 102
EMB_FORMAT_STX
 embroidery.h, 102
EMB_FORMAT_SVG
 embroidery.h, 102
EMB_FORMAT_T01
 embroidery.h, 103
EMB_FORMAT_T09
 embroidery.h, 103

EMB_FORMAT_TAP
embroidery.h, 103
EMB_FORMAT_THR
embroidery.h, 103
EMB_FORMAT_TXT
embroidery.h, 103
EMB_FORMAT_U00
embroidery.h, 103
EMB_FORMAT_U01
embroidery.h, 104
EMB_FORMAT_VIP
embroidery.h, 104
EMB_FORMAT_VP3
embroidery.h, 104
EMB_FORMAT_XXX
embroidery.h, 104
EMB_FORMAT_ZSK
embroidery.h, 104
emb_identify_format
embroidery.h, 120
formats.c, 259
EMB_IMAGE
embroidery.h, 104
EMB_INT16_BIG
embroidery_internal.h, 176
EMB_INT16_LITTLE
embroidery_internal.h, 176
EMB_INT32_BIG
embroidery_internal.h, 176
EMB_INT32_LITTLE
embroidery_internal.h, 177
EMB_LINE
embroidery.h, 105
EMB_LITTLE_ENDIAN
embroidery_internal.h, 177
EMB_MAX
embroidery_internal.h, 177
EMB_MAX_LAYERS
embroidery.h, 105
EMB_MIN
embroidery_internal.h, 177
emb_optOut
embroidery_internal.h, 194
main.c, 287
EMB_PATH
embroidery.h, 105
EMB_POINT
embroidery.h, 105
EMB_POLYGON
embroidery.h, 105
EMB_POLYLINE
embroidery.h, 105
EMB_PUBLIC
embroidery.h, 106
emb_readline
embroidery_internal.h, 194
main.c, 287
EMB_RECT

embroidery.h, 106
emb_round
embroidery.h, 120
EMB_SPLINE
embroidery.h, 106
EMB_STITCH
embroidery.h, 106
EMB_TEXT_MULTI
embroidery.h, 106
EMB_TEXT_SINGLE
embroidery.h, 106
EMB_THREAD
embroidery.h, 107
EMB_VECTOR
embroidery.h, 107
emb_verbose
embroidery.h, 145
main.c, 294
EmbAlignedDim
embroidery.h, 114
EmbAlignedDim_, 22
position, 22
EmbAngularDim
embroidery.h, 114
EmbAngularDim_, 23
position, 23
EmbArc
embroidery.h, 114
EmbArc_, 23
end, 24
mid, 24
start, 24
embArc_clockwise
embroidery.h, 120
embArc_init
embroidery.h, 121
embArc_print
main.c, 287
EmbArcLengthDim
embroidery.h, 114
EmbArcLengthDim_, 24
position, 25
EmbArray
embroidery.h, 115
EmbArray_, 25
count, 25
geometry, 26
length, 26
stitch, 26
thread, 26
type, 26
embArray_addArc
array.c, 71
embroidery.h, 121
embArray_addCircle
array.c, 72
embroidery.h, 121
embArray_addEllipse

array.c, 72
 embroidery.h, 121
embArray_addFlag
 array.c, 72
 embroidery.h, 121
embArray_addLine
 array.c, 72
 embroidery.h, 121
embArray_addPath
 array.c, 72
 embroidery.h, 122
embArray_addPoint
 array.c, 73
 embroidery.h, 122
embArray_addPolygon
 array.c, 73
 embroidery.h, 122
embArray_addPolyline
 array.c, 73
 embroidery.h, 122
embArray_addRect
 array.c, 73
 embroidery.h, 122
embArray_addStitch
 array.c, 73
 embroidery.h, 123
embArray_addThread
 embroidery.h, 123
embArray_addVector
 array.c, 74
 embroidery.h, 123
embArray_copy
 array.c, 74
 embroidery.h, 123
embArray_create
 array.c, 74
 embroidery.h, 123
embArray_free
 array.c, 74
 embroidery.h, 124
embArray_resize
 array.c, 74
 embroidery.h, 124
EmbBezier
 embroidery.h, 115
EmbBezier_, 27
 control1, 27
 control2, 27
 end, 27
 start, 27
EmbBlock
 embroidery.h, 115
EmbBlock_, 28
 position, 28
EmbCircle
 embroidery.h, 115
EmbCircle_, 28
 center, 29
 radius, 29
embCircle_init
 embroidery.h, 124
EmbColor
 embroidery.h, 115
EmbColor_, 29
 b, 30
 g, 30
 r, 30
embColor_create
 embroidery.h, 124
embColor_distance
 embroidery.h, 124
 main.c, 288
embColor_fromHexStr
 embroidery.h, 124
 encoding.c, 232
embColor_make
 embroidery.h, 125
embColor_read
 embroidery_internal.h, 194
 main.c, 288
embColor_write
 embroidery_internal.h, 194
 main.c, 288
embConstantPi
 embroidery.h, 146
 main.c, 294
EmbDiameterDim
 embroidery.h, 115
EmbDiameterDim_, 30
 position, 30
EmbEllipse
 embroidery.h, 115
EmbEllipse_, 31
 center, 31
 radius, 31
 rotation, 31
embEllipse_area
 embroidery.h, 125
embEllipse_diameterX
 embroidery.h, 125
embEllipse_diameterY
 embroidery.h, 125
embEllipse_height
 embroidery.h, 125
embEllipse_init
 embroidery.h, 125
embEllipse_make
 embroidery.h, 126
embEllipse_perimeter
 embroidery.h, 126
embEllipse_width
 embroidery.h, 126
EmbFlag
 embroidery.h, 116
embFormat_getExtension
 formats.c, 259

EMBFORMAT_MAXDESC
embroidery.h, 107
EMBFORMAT_MAXEXT
embroidery.h, 107
EMBFORMAT_OBJECTONLY
embroidery.h, 107
EMBFORMAT_STCHANDOBJ
embroidery.h, 107
EMBFORMAT_STITCHONLY
embroidery.h, 108
EMBFORMAT_UNSUPPORTED
embroidery.h, 108
EmbFormatList
embroidery.h, 116
EmbFormatList_, 32
check_for_color_file, 32
color_only, 32
description, 32
extension, 33
reader_state, 33
type, 33
write_external_color_file, 33
writer_state, 33
EmbGeometry
embroidery.h, 116
EmbGeometry_, 34
arc, 34
circle, 34
color, 34
ellipse, 35
flag, 35
line, 35
lineType, 35
object, 35
path, 35
point, 36
polygon, 36
polyline, 36
rect, 36
spline, 36
stitch, 36
thread, 37
type, 37
vector, 37
embGeometry_boundingRect
embroidery.h, 126
geometry.c, 270
embGeometry_free
embroidery.h, 126
geometry.c, 270
embGeometry_init
embroidery.h, 126
geometry.c, 271
embGeometry_move
embroidery.h, 127
geometry.c, 271
embGeometry_vulcanize
embroidery.h, 127
geometry.c, 271
EmblImage
embroidery.h, 116
EmblImage_, 37
data, 38
dimensions, 38
height, 38
name, 38
path, 38
position, 38
width, 39
emblImage_create
embroidery.h, 127
emblImage_free
embroidery.h, 127
emblImage_read
embroidery.h, 127
emblImage_write
embroidery.h, 127
EmblInfiniteLine
embroidery.h, 116
EmblInfiniteLine_, 39
position, 39
emblInt_read
embroidery_internal.h, 195
encoding.c, 232
emblInt_write
embroidery_internal.h, 195
encoding.c, 233
EmbLayer
embroidery.h, 116
EmbLayer_, 40
geometry, 40
name, 40
EmbLeaderDim
embroidery.h, 116
EmbLeaderDim_, 40
position, 41
EmbLine
embroidery.h, 117
EmbLine_, 41
color, 41
end, 41
lineType, 42
start, 42
embLine_intersectionPoint
embroidery.h, 128
embLine_make
embroidery.h, 128
embLine_normalVector
embroidery.h, 128
EmbLinearDim
embroidery.h, 117
EmbLinearDim_, 42
position, 42
EmbOrdinateDim
embroidery.h, 117
EmbOrdinateDim_, 43

position, 43
EmbPath
 embroidery.h, 117
EmbPath_, 43
 color, 44
 flagList, 44
 lineType, 44
 pointList, 44
EmbPattern
 embroidery.h, 117
EmbPattern_, 45
 currentColorIndex, 45
 dstJumpsPerTrim, 45
 geometry, 45
 home, 45
 hoop_height, 46
 hoop_width, 46
 layer, 46
 stitch_list, 46
 thread_list, 46
embPattern_addCircleAbs
 embroidery.h, 128
 pattern.c, 316
embPattern_addEllipseAbs
 embroidery.h, 128
 pattern.c, 316
embPattern_addLineAbs
 embroidery.h, 129
 pattern.c, 317
embPattern_addPathAbs
 embroidery.h, 129
 pattern.c, 317
embPattern_addPointAbs
 embroidery.h, 129
 pattern.c, 317
embPattern_addPolygonAbs
 embroidery.h, 129
 pattern.c, 317
embPattern_addPolylineAbs
 embroidery.h, 130
embPattern_addPolylineObjectAbs
 pattern.c, 318
embPattern_addRectAbs
 embroidery.h, 130
 pattern.c, 318
embPattern_addStitchAbs
 embroidery.h, 130
 pattern.c, 318
embPattern_addStitchRel
 embroidery.h, 130
 pattern.c, 318
embPattern_addThread
 embroidery.h, 131
 pattern.c, 319
embPattern_calcBoundingBox
 embroidery.h, 131
 pattern.c, 319
embPattern_center
 embroidery.h, 131
 pattern.c, 319
embPattern_changeColor
 embroidery.h, 131
 pattern.c, 319
embPattern_color_count
 embroidery.h, 131
 pattern.c, 319
embPattern_combine
 embroidery.h, 132
 fill.c, 240
embPattern_combineJumpStitches
 embroidery.h, 132
 pattern.c, 320
embPattern_convertGeometry
 embroidery.h, 132
 fill.c, 240
embPattern_copyPolylinesToStitch_list
 pattern.c, 320
embPattern_copyPolylinesToStitchList
 embroidery.h, 132
embPattern_copystitch_listToPolylines
 pattern.c, 320
embPattern_copyStitchListToPolylines
 embroidery.h, 132
embPattern_correctForMaxStitchLength
 embroidery.h, 132
 pattern.c, 320
embPattern_create
 embroidery.h, 133
 pattern.c, 320
embPattern_crossstitch
 embroidery.h, 133
 fill.c, 241
embPattern_designDetails
 embroidery.h, 133
 pattern.c, 321
embPattern_end
 embroidery.h, 133
 pattern.c, 321
embPattern_fixColorCount
 embroidery.h, 133
 pattern.c, 321
embPattern_flip
 embroidery.h, 134
 pattern.c, 321
embPattern_flipHorizontal
 embroidery.h, 134
 pattern.c, 321
embPattern_flipVertical
 embroidery.h, 134
 pattern.c, 322
embPattern_free
 embroidery.h, 134
 pattern.c, 322
embPattern_hideStitchesOverLength
 embroidery.h, 134
 pattern.c, 322

embPattern_horizontal_fill
embroidery.h, 135
fill.c, 241
embPattern_jumpStitches
embroidery.h, 135
pattern.c, 322
embPattern_lengthHistogram
embroidery.h, 135
pattern.c, 322
embPattern_loadExternalColorFile
embroidery.h, 135
pattern.c, 323
embPattern_maximumStitchLength
embroidery.h, 135
pattern.c, 323
embPattern_minimumStitchLength
embroidery.h, 136
pattern.c, 323
embPattern_movePolylinesToStitch_list
pattern.c, 323
embPattern_movePolylinesToStitchList
embroidery.h, 136
embPattern_movestitch_listToPolylines
pattern.c, 323
embPattern_moveStitchListToPolylines
embroidery.h, 136
embPattern_read
embroidery.h, 136
formats.c, 259
embPattern_readAuto
embroidery.h, 136
formats.c, 259
embPattern_realStitches
embroidery.h, 136
pattern.c, 324
embPattern_render
embroidery.h, 137
embPattern_scale
embroidery.h, 137
pattern.c, 324
embPattern_simulate
embroidery.h, 137
embPattern_stitchArc
fill.c, 241
embPattern_stitchCircle
fill.c, 241
embPattern_stitchEllipse
fill.c, 241
embPattern_stitchPath
fill.c, 242
embPattern_stitchPolygon
fill.c, 242
embPattern_stitchPolyline
fill.c, 242
embPattern_stitchRect
fill.c, 242
embPattern_stitchText
fill.c, 243
embPattern_totalStitchLength
embroidery.h, 137
pattern.c, 324
embPattern_trimStitches
embroidery.h, 137
pattern.c, 324
embPattern_write
embroidery.h, 137
formats.c, 259
embPattern_writeAuto
embroidery.h, 138
formats.c, 260
EmbPoint
embroidery.h, 117
EmbPoint_, 47
color, 47
lineType, 47
position, 47
EmbPolygon
embroidery.h, 117
embPolygon_reduceByDistance
fill.c, 243
embPolygon_reduceByNth
fill.c, 243
EmbPolyline
embroidery.h, 118
EmbRadiusDim
embroidery.h, 118
EmbRadiusDim_, 48
position, 48
EmbRay
embroidery.h, 118
EmbRay_, 48
position, 48
EmbReal
embroidery.h, 118
EmbRect
embroidery.h, 118
EmbRect_, 49
bottom, 49
left, 49
radius, 49
right, 50
rotation, 50
top, 50
embRect_area
embroidery.h, 138
embRect_init
embroidery.h, 138
embroidery.h
_dxfColorTable, 145
Arc_Polyester, 92
Arc_Rayon, 92
black_thread, 145
CHUNK_SIZE, 93
CoatsAndClark_Rayon, 93
convert, 120
degrees, 120

dx_f_color, 93
EMB_ARC, 93
EMB_ARRAY, 93
EMB_CIRCLE, 93
EMB_DIM_DIAMETER, 94
EMB_DIM_LEADER, 94
EMB_ELLIPSE, 94
emb_error, 145
EMB_FLAG, 94
EMB_FORMAT_100, 94
EMB_FORMAT_100, 94
EMB_FORMAT_ART, 95
EMB_FORMAT_BMC, 95
EMB_FORMAT_BRO, 95
EMB_FORMAT_CND, 95
EMB_FORMAT_COL, 95
EMB_FORMAT_CSD, 95
EMB_FORMAT_CSV, 96
EMB_FORMAT_DAT, 96
EMB_FORMATDEM, 96
EMB_FORMAT_DSB, 96
EMB_FORMAT_DST, 96
EMB_FORMAT_DSZ, 96
EMB_FORMAT_DXF, 97
EMB_FORMAT_EDR, 97
EMB_FORMAT_EMD, 97
EMB_FORMAT_EXP, 97
EMB_FORMAT_EXY, 97
EMB_FORMAT_EYS, 97
EMB_FORMAT_FXY, 98
EMB_FORMAT_GC, 98
EMB_FORMAT_GNC, 98
EMB_FORMAT_GT, 98
EMB_FORMAT_HUS, 98
EMB_FORMAT_INB, 98
EMB_FORMAT_INF, 99
EMB_FORMAT_JEF, 99
EMB_FORMAT_KSM, 99
EMB_FORMAT_MAX, 99
EMB_FORMATMIT, 99
EMB_FORMAT_NEW, 99
EMB_FORMAT_OFM, 100
EMB_FORMAT_PCD, 100
EMB_FORMAT_PCM, 100
EMB_FORMAT_PCQ, 100
EMB_FORMAT_PCS, 100
EMB_FORMAT_PEC, 100
EMB_FORMAT_PEL, 101
EMB_FORMAT_PEM, 101
EMB_FORMAT_PES, 101
EMB_FORMAT_PHB, 101
EMB_FORMAT_PHC, 101
EMB_FORMAT_PLT, 101
EMB_FORMAT_RGB, 102
EMB_FORMAT_SEW, 102
EMB_FORMAT_SHV, 102
EMB_FORMAT_SST, 102
EMB_FORMAT_STX, 102
EMB_FORMAT_SVG, 102
EMB_FORMAT_T01, 103
EMB_FORMAT_T09, 103
EMB_FORMAT_TAP, 103
EMB_FORMAT_THR, 103
EMB_FORMAT_TXT, 103
EMB_FORMAT_U00, 103
EMB_FORMAT_U01, 104
EMB_FORMAT_VIP, 104
EMB_FORMAT_VP3, 104
EMB_FORMAT_XXX, 104
EMB_FORMAT_ZSK, 104
emb_identify_format, 120
EMB_IMAGE, 104
EMB_LINE, 105
EMB_MAX_LAYERS, 105
EMB_PATH, 105
EMB_POINT, 105
EMB_POLYGON, 105
EMB_POLYLINE, 105
EMB_PUBLIC, 106
EMB_RECT, 106
emb_round, 120
EMB_SPLINE, 106
EMB_STITCH, 106
EMB_TEXT_MULTI, 106
EMB_TEXT_SINGLE, 106
EMB_THREAD, 107
EMB_VECTOR, 107
emb_verbose, 145
EmbAlignedDim, 114
EmbAngularDim, 114
EmbArc, 114
embArc_clockwise, 120
embArc_init, 121
EmbArcLengthDim, 114
EmbArray, 115
embArray_addArc, 121
embArray_addCircle, 121
embArray_addEllipse, 121
embArray_addFlag, 121
embArray_addLine, 121
embArray_addPath, 122
embArray_addPoint, 122
embArray_addPolygon, 122
embArray_addPolyline, 122
embArray_addRect, 122
embArray_addStitch, 123
embArray_addThread, 123
embArray_addVector, 123
embArray_copy, 123
embArray_create, 123
embArray_free, 124
embArray_resize, 124
EmbBezier, 115
EmbBlock, 115
EmbCircle, 115
embCircle_init, 124

EmbColor, 115
embColor_create, 124
embColor_distance, 124
embColor_fromHexStr, 124
embColor_make, 125
embConstantPi, 146
EmbDiameterDim, 115
EmbEllipse, 115
embEllipse_area, 125
embEllipse_diameterX, 125
embEllipse_diameterY, 125
embEllipse_height, 125
embEllipse_init, 125
embEllipse_make, 126
embEllipse_perimeter, 126
embEllipse_width, 126
EmbFlag, 116
EMBFORMAT_MAXDESC, 107
EMBFORMAT_MAXEXT, 107
EMBFORMAT_OBJECTONLY, 107
EMBFORMAT_STCHANDOBJ, 107
EMBFORMAT_STITCHONLY, 108
EMBFORMAT_UNSUPPORTED, 108
EmbFormatList, 116
EmbGeometry, 116
embGeometry_boundingRect, 126
embGeometry_free, 126
embGeometry_init, 126
embGeometry_move, 127
embGeometry_vulcanize, 127
EmblImage, 116
emblImage_create, 127
emblImage_free, 127
emblImage_read, 127
emblImage_write, 127
EmblInfiniteLine, 116
EmbLayer, 116
EmbLeaderDim, 116
EmbLine, 117
embLine_intersectionPoint, 128
embLine_make, 128
embLine_normalVector, 128
EmbLinearDim, 117
EmbOrdinateDim, 117
EmbPath, 117
EmbPattern, 117
embPattern_addCircleAbs, 128
embPattern_addEllipseAbs, 128
embPattern_addLineAbs, 129
embPattern_addPathAbs, 129
embPattern_addPointAbs, 129
embPattern_addPolygonAbs, 129
embPattern_addPolylineAbs, 130
embPattern_addRectAbs, 130
embPattern_addStitchAbs, 130
embPattern_addStitchRel, 130
embPattern_addThread, 131
embPattern_calcBoundingBox, 131
embPattern_center, 131
embPattern_changeColor, 131
embPattern_color_count, 131
embPattern_combine, 132
embPattern_combineJumpStitches, 132
embPattern_convertGeometry, 132
embPattern_copyPolylinesToStitchList, 132
embPattern_copyStitchListToPolylines, 132
embPattern_correctForMaxStitchLength, 132
embPattern_create, 133
embPattern_crossstitch, 133
embPattern_designDetails, 133
embPattern_end, 133
embPattern_fixColorCount, 133
embPattern_flip, 134
embPattern_flipHorizontal, 134
embPattern_flipVertical, 134
embPattern_free, 134
embPattern_hideStitchesOverLength, 134
embPattern_horizontal_fill, 135
embPattern_jumpStitches, 135
embPattern_lengthHistogram, 135
embPattern_loadExternalColorFile, 135
embPattern_maximumStitchLength, 135
embPattern_minimumStitchLength, 136
embPattern_movePolylinesToStitchList, 136
embPattern_moveStitchListToPolylines, 136
embPattern_read, 136
embPattern_readAuto, 136
embPattern_realStitches, 136
embPattern_render, 137
embPattern_scale, 137
embPattern_simulate, 137
embPattern_totalStitchLength, 137
embPattern_trimStitches, 137
embPattern_write, 137
embPattern_writeAuto, 138
EmbPoint, 117
EmbPolygon, 117
EmbPolyline, 118
EmbRadiusDim, 118
EmbRay, 118
EmbReal, 118
EmbRect, 118
embRect_area, 138
embRect_init, 138
EmbSatinOutline, 118
embSatinOutline_generateSatinOutline, 138
embSatinOutline_renderStitches, 138
EmbSpline, 118
EmbStitch, 119
EmbTextMulti, 119
EmbTextSingle, 119
EmbThread, 119
embThread_findNearestColor, 139
embThread_findNearestThread, 139
embThread_getRandom, 139
EmbTime, 119

embTime_initNow, 140
 embTime_time, 140
 EmbVector, 119
 embVector_add, 140
 embVector_angle, 140
 embVector_average, 140
 embVector_cross, 141
 embVector_distance, 141
 embVector_dot, 141
 embVector_length, 141
 embVector_multiply, 141
 embVector_normalize, 141
 embVector_relativeX, 142
 embVector_relativeY, 142
 embVector_subtract, 142
 embVector_transpose_product, 142
 embVector_unit, 142
 END, 108
 Exquisite_Polyester, 108
 formatTable, 146
 Fufu_Polyester, 108
 Fufu_Rayon, 108
 full_test_matrix, 142
 getArcCenter, 143
 getArcDataFromBulge, 143
 getCircleCircleIntersections, 143
 getCircleTangentPoints, 143
 Hemingworth_Polyester, 109
 hilbert_curve, 143
 hus_thread, 109
 husThreads, 146
 Isacord_Polyester, 109
 Isafil_Rayon, 109
 jef_thread, 109
 jefThreads, 146
 JUMP, 109
 L_system, 119
 LIBEMBROIDERY_EMBEDDED_VERSION, 110
 lindenmayer_system, 144
 Madeira_Polyester, 110
 Madeira_Rayon, 110
 Marathon_Polyester, 110
 Marathon_Rayon, 110
 MAX_STITCHES, 110
 MAX_THREADS, 111
 Metro_Polyester, 111
 NORMAL, 111
 numberFormats, 111
 Pantone, 111
 pcm_thread, 111
 pcmThreads, 146
 pec_thread, 112
 pecThreadCount, 146
 pecThreads, 147
 radians, 144
 report, 144
 RobisonAnton_Polyester, 112
 RobisonAnton_Rayon, 112
 SEQUIN, 112
 shv_thread, 112
 shvThreadCount, 147
 shvThreads, 147
 Sigma_Polyester, 112
 STOP, 113
 Sulky_Rayon, 113
 SVG_Colors, 113
 testMain, 144
 thread_color, 120
 ThreadArt_Polyester, 113
 ThreadArt_Rayon, 113
 threadColor, 144
 threadColorName, 144
 threadColorNum, 145
 ThreaDelight_Polyester, 113
 TRIM, 114
 vipDecodingTable, 147
 Z102_Isacord_Polyester, 114
embroidery_internal.h
 bcf_difat_create, 188
 bcf_directory, 185
 bcf_directory_entry, 185
 bcf_directory_free, 188
 bcf_file, 186
 bcf_file_difat, 186
 bcf_file_difat_free, 188
 bcf_file_fat, 186
 bcf_file_fat_free, 189
 bcf_file_free, 189
 bcf_file_header, 186
 bcfFile_read, 189
 bcfFileFat_create, 189
 bcfFileHeader_isValid, 189
 bcfFileHeader_read, 189
 binaryReadString, 190
 binaryReadUnicodeString, 190
 binaryWriteUInt, 190
 BULGETOCONTROL, 162
 BULGETOEND, 163
 check_header_present, 190
 CompoundFileDirectory, 190
 CompoundFileDirectoryEntry, 191
 CompoundFileSector_DIFAT_Sector, 163
 CompoundFileSector_EndOfChain, 163
 CompoundFileSector_FAT_Sector, 163
 CompoundFileSector_FreeSector, 163
 CompoundFileSector_MaxRegSector, 163
 CompoundFileStreamId_MaxRegularStreamId,
 164
 CompoundFileStreamId_NoStream, 164
 compress, 186
 compress_get_bits, 191
 compress_get_position, 191
 compress_get_token, 191
 compress_load_block, 191
 compress_load_character_huffman, 192
 compress_load_character_length_huffman, 192

compress_load_distance_huffman, 192
compress_pop, 192
compress_read_variable_length, 192
copy_trim, 193
create_test_file_1, 193
create_test_file_2, 193
create_test_file_3, 193
CSV_EXPECT, 187
CSV_EXPECT_COMMA, 188
CSV_EXPECT_NULL, 188
CSV_EXPECT_QUOTE1, 188
CSV_EXPECT_QUOTE2, 188
CSV_MODE, 188
CSV_MODE_COMMENT, 188
CSV_MODE_NULL, 188
CSV_MODE_STITCH, 188
CSV_MODE_THREAD, 188
CSV_MODE_VARIABLE, 188
CUBICTOCONTROL1, 164
CUBICTOCONTROL2, 164
CUBICTOEND, 164
decode_t01_record, 193
decode_tajima_ternary, 193
decodeNewStitch, 194
DXF_VERSION_2000, 165
DXF_VERSION_2002, 165
DXF_VERSION_2004, 165
DXF_VERSION_2006, 165
DXF_VERSION_2007, 165
DXF_VERSION_2009, 165
DXF_VERSION_2010, 166
DXF_VERSION_2013, 166
DXF_VERSION_R10, 166
DXF_VERSION_R11, 166
DXF_VERSION_R12, 166
DXF_VERSION_R13, 166
DXF_VERSION_R14, 167
DXF_VERSION_R15, 167
DXF_VERSION_R18, 167
DXF_VERSION_R21, 167
DXF_VERSION_R24, 167
DXF_VERSION_R27, 167
ELEMENT_A, 168
ELEMENT_ANIMATE, 168
ELEMENT_ANIMATECOLOR, 168
ELEMENT_ANIMATEMOTION, 168
ELEMENT_ANIMATETRANSFORM, 168
ELEMENT_ANIMATION, 168
ELEMENT_AUDIO, 169
ELEMENT_CIRCLE, 169
ELEMENT_DEFS, 169
ELEMENT_DESC, 169
ELEMENT_DISCARD, 169
ELEMENT_ELLIPSE, 169
ELEMENT_FONT, 170
ELEMENT_FONT_FACE, 170
ELEMENT_FONT_FACE_SRC, 170
ELEMENT_FONT_FACE_URI, 170
ELEMENT_FOREIGN_OBJECT, 170
ELEMENT_G, 170
ELEMENT_GLYPH, 171
ELEMENT_HANDLER, 171
ELEMENT_HKERN, 171
ELEMENT_IMAGE, 171
ELEMENT_LINE, 171
ELEMENT_LINEAR_GRADIENT, 171
ELEMENT_LISTENER, 172
ELEMENT_METADATA, 172
ELEMENT_MISSING_GLYPH, 172
ELEMENT_MPATH, 172
ELEMENT_PATH, 172
ELEMENT_POLYGON, 172
ELEMENT_POLYLINE, 173
ELEMENT_PREFETCH, 173
ELEMENT_RADIAL_GRADIENT, 173
ELEMENT_RECT, 173
ELEMENT_SCRIPT, 173
ELEMENT_SET, 173
ELEMENT_SOLID_COLOR, 174
ELEMENT_STOP, 174
ELEMENT_SVG, 174
ELEMENT_SWITCH, 174
ELEMENT_TBREAK, 174
ELEMENT_TEXT, 174
ELEMENT_TEXT_AREA, 175
ELEMENT_TITLE, 175
ELEMENT_TSPAN, 175
ELEMENT_USE, 175
ELEMENT_VIDEO, 175
ELEMENT_XML, 175
ELLIPSETOEND, 176
ELLIPSETORAD, 176
EMB_BIG_ENDIAN, 176
EMB_INT16_BIG, 176
EMB_INT16_LITTLE, 176
EMB_INT32_BIG, 176
EMB_INT32_LITTLE, 177
EMB_LITTLE_ENDIAN, 177
EMB_MAX, 177
EMB_MIN, 177
emb_optOut, 194
emb_readline, 194
embColor_read, 194
embColor_write, 194
emblnt_read, 195
emblnt_write, 195
encode_t01_record, 195
encode_tajima_ternary, 195
ENDIAN_HOST, 177
entriesInDifatSector, 196
fpad, 196
fread_int16, 196
fread_uint16, 196
GetFile, 196
GREEN_TERM_COLOR, 177
HOOP_110X110, 178

HOOP_126X110, 178
HOOP_140X200, 178
HOOP_230X200, 178
HOOP_50X50, 178
huffman, 186
huffman_build_table, 197
huffman_table_lookup, 197
hus_compress, 197
hus_decompress, 197
imageWithFrame, 224
LINETO, 178
loadFatFromSector, 197
mitDecodeStitch, 198
mitEncodeStitch, 198
MOVETO, 179
N_PES VERSIONS, 179
numberOfEntriesInDifatSector, 198
ObjectTypeRootEntry, 179
ObjectTypeStorage, 179
ObjectTypeStream, 179
ObjectTypeUnknown, 180
PES0001, 180
PES0020, 180
PES0022, 180
PES0030, 180
PES0040, 181
PES0050, 181
PES0055, 181
PES0056, 181
PES0060, 181
PES0070, 181
PES0080, 182
PES0090, 182
PES0100, 182
pfaffDecode, 198
pfaffEncode, 198
printArcResults, 199
QUADTOCONTROL, 182
QUADTOEND, 182
read100, 199
read10o, 199
readArt, 199
readBmc, 199
readBro, 200
readCnd, 200
readCol, 200
readCsd, 200
readCsv, 200
readDat, 200
readDem, 201
readDescriptions, 201
readDsb, 201
readDst, 201
readDsz, 201
readDxf, 201
readEdr, 202
readEmd, 202
readExp, 202
readExy, 202
readEys, 202
readFeatherPatterns, 202
readFullSector, 203
readFxy, 203
readGc, 203
readGnc, 203
readGt, 203
readHoopName, 204
readHus, 204
readImageString, 204
readInb, 204
readInf, 204
readJef, 204
readKsm, 204
readMax, 205
readMit, 205
readMotifPatterns, 205
readNew, 205
readNextSector, 205
readOfm, 205
readPcd, 206
readPcm, 206
readPcq, 206
readPcs, 206
readPec, 206
readPecStitches, 206
readPel, 207
readPem, 207
readPes, 207
readPESHeaderV10, 207
readPESHeaderV5, 207
readPESHeaderV6, 207
readPESHeaderV7, 208
readPESHeaderV8, 208
readPESHeaderV9, 208
readPhb, 208
readPhc, 208
readPlt, 208
readProgrammableFills, 209
readRgb, 209
readSew, 209
readShv, 209
readSst, 209
readStx, 209
readSvg, 210
readT01, 210
readT09, 210
readTap, 210
readThr, 210
readThreads, 210
readTxt, 211
readU00, 211
readU01, 211
readVip, 211
readVp3, 211
readXxx, 211
readZsk, 212

RED_TERM_COLOR, 182
RESET_TERM_COLOR, 183
safe_free, 212
stringInArray, 212
StxThread, 186
SubDescriptor, 186
SVG_ATTRIBUTE, 183
SVG_CATCH_ALL, 183
SVG_CREATOR_EMBROIDERMODDER, 183
SVG_CREATOR_ILLUSTRATOR, 183
SVG_CREATOR_INKSCAPE, 183
SVG_CREATOR_NULL, 184
SVG_ELEMENT, 184
SVG_EXPECT_ATTRIBUTE, 184
SVG_EXPECT_ELEMENT, 184
SVG_EXPECT_NULL, 184
SVG_EXPECT_VALUE, 184
SVG_MEDIA_PROPERTY, 185
SVG_NULL, 185
SVG_PROPERTY, 185
SvgAttribute, 187
testEmbCircle, 212
testEmbCircle_2, 212
testEmbFormat, 213
testGeomArc, 213
testTangentPoints, 213
testThreadColor, 213
ThredExtension, 187
ThredHeader, 187
VipHeader, 187
vp3Hoop, 187
write100, 213
write10o, 213
write_24bit, 214
writeArt, 214
writeBmc, 214
writeBro, 214
writeCnd, 214
writeCol, 214
writeCsd, 215
writeCsv, 215
writeDat, 215
writeDem, 215
writeDsb, 215
writeDst, 215
writeDsz, 216
writeDxf, 216
writeEdr, 216
writeEmd, 216
writeExp, 216
writeExy, 216
writeEys, 217
writeFxy, 217
writeGc, 217
writeGnc, 217
writeGt, 217
writeHus, 217
writeInb, 218
writeInf, 218
writeJef, 218
writeKsm, 218
writeMax, 218
writeMit, 218
writeNew, 219
writeOfm, 219
writePcd, 219
writePcm, 219
writePcq, 219
writePcs, 219
writePec, 220
writePecStitches, 220
writePel, 220
writePem, 220
writePes, 220
writePhb, 220
writePhc, 221
writePlt, 221
writeRgb, 221
writeSew, 221
writeShv, 221
writeSst, 221
writeStx, 222
writeSvg, 222
writeT01, 222
writeT09, 222
writeTap, 222
writeThr, 222
writeTxt, 223
writeU00, 223
writeU01, 223
writeVip, 223
writeVp3, 223
writeXxx, 223
writeZsk, 224
YELLOW_TERM_COLOR, 185
EmbSatinOutline
 embroidery.h, 118
EmbSatinOutline_
 length, 51
 side1, 51
 side2, 51
 embSatinOutline_generateSatinOutline
 embroidery.h, 138
 main.c, 288
 embSatinOutline_renderStitches
 embroidery.h, 138
 main.c, 288
 EmbSpline
 embroidery.h, 118
 EmbSpline_
 51
 beziers, 51
 EmbStitch
 embroidery.h, 119
 EmbStitch_
 52
 color, 52
 flags, 52

x, 52
 y, 53
EmbTextMulti
 embroidery.h, 119
EmbTextMulti_, 53
 position, 53
 text, 53
EmbTextSingle
 embroidery.h, 119
EmbTextSingle_, 54
 position, 54
 text, 54
EmbThread
 embroidery.h, 119
EmbThread_, 55
 catalogNumber, 55
 color, 55
 description, 55
embThread_findNearestColor
 embroidery.h, 139
 main.c, 289
embThread_findNearestThread
 embroidery.h, 139
 main.c, 289
embThread_getRandom
 embroidery.h, 139
 main.c, 289
EmbTime
 embroidery.h, 119
EmbTime_, 56
 day, 56
 hour, 56
 minute, 56
 month, 56
 second, 57
 year, 57
embTime_initNow
 embroidery.h, 140
 main.c, 290
embTime_time
 embroidery.h, 140
 main.c, 290
EmbVector
 embroidery.h, 119
EmbVector_, 57
 x, 57
 y, 58
embVector_add
 embroidery.h, 140
embVector_angle
 embroidery.h, 140
embVector_average
 embroidery.h, 140
embVector_cross
 embroidery.h, 141
embVector_distance
 embroidery.h, 141
embVector_dot
 embroidery.h, 141
embVector_length
 embroidery.h, 141
embVector_multiply
 embroidery.h, 141
embVector_normalize
 embroidery.h, 141
embVector_print
 main.c, 290
embVector_relativeX
 embroidery.h, 142
embVector_relativeY
 embroidery.h, 142
embVector_subtract
 embroidery.h, 142
embVector_transpose_product
 embroidery.h, 142
embVector_unit
 embroidery.h, 142
encode_t01_record
 embroidery_internal.h, 195
 encoding.c, 233
encode_tajima_ternary
 embroidery_internal.h, 195
 encoding.c, 233
encoding.c
 decode_t01_record, 232
 decode_tajima_ternary, 232
 decodeNewStitch, 232
 embColor_fromHexStr, 232
 emblnt_read, 232
 emblnt_write, 233
 encode_t01_record, 233
 encode_tajima_ternary, 233
 mitDecodeStitch, 233
 mitEncodeStitch, 233
 pfaffDecode, 234
 pfaffEncode, 234
 reverse_byte_order, 234
 write_24bit, 234
END
 embroidery.h, 108
end
 EmbArc_, 24
 EmbBezier_, 27
 EmbLine_, 41
ENDIAN_HOST
 embroidery_internal.h, 177
entriesInDifatSector
 embroidery_internal.h, 196
 main.c, 290
Exquisite_Polyester
 embroidery.h, 108
extension
 EmbFormatList_, 33
fat
 _bcf_file, 9
fatEntries

_bcf_file_fat, 11
fatEntryCount
 _bcf_file_fat, 11
fatSectorCount
 _bcf_file_difat, 10
fatSectorEntries
 _bcf_file_difat, 10
fill.c
 dragon_curve, 240
 embPattern_combine, 240
 embPattern_convertGeometry, 240
 embPattern_crossstitch, 241
 embPattern_horizontal_fill, 241
 embPattern_stitchArc, 241
 embPattern_stitchCircle, 241
 embPattern_stitchEllipse, 241
 embPattern_stitchPath, 242
 embPattern_stitchPolygon, 242
 embPattern_stitchPolyline, 242
 embPattern_stitchRect, 242
 embPattern_stitchText, 243
 embPolygon_reduceByDistance, 243
 embPolygon_reduceByNth, 243
 generate_dragon_curve, 243
 hilbert_curve, 243
 hilbert_curve_l_system, 244
 lindenmayer_system, 244
 rules, 244
firstDifatSectorLocation
 _bcf_file_header, 13
firstDirectorySectorLocation
 _bcf_file_header, 13
firstMiniFATSectorLocation
 _bcf_file_header, 13
flag
 EmbGeometry_, 35
FLAG_CIRCLE
 main.c, 278
FLAG_CIRCLE_SHORT
 main.c, 278
FLAG_COMBINE
 main.c, 279
FLAG_CROSS_STITCH
 main.c, 279
FLAG_ELLIPSE
 main.c, 279
FLAG_ELLIPSE_SHORT
 main.c, 279
FLAG_FILL
 main.c, 279
FLAG_FILL_SHORT
 main.c, 279
FLAG_FORMATS
 main.c, 280
FLAG_FORMATS_SHORT
 main.c, 280
FLAG_FULL_TEST_SUITE
 main.c, 280
FLAG_HELP
 main.c, 280
FLAG_HELP_SHORT
 main.c, 280
FLAG_HILBERT_CURVE
 main.c, 280
FLAG_LINE
 main.c, 281
FLAG_LINE_SHORT
 main.c, 281
FLAG_POLYGON
 main.c, 281
FLAG_POLYGON_SHORT
 main.c, 281
FLAG_POLYLINE
 main.c, 281
FLAG_POLYLINE_SHORT
 main.c, 281
FLAG QUIET
 main.c, 282
FLAG QUIET_SHORT
 main.c, 282
FLAG_RENDER
 main.c, 282
FLAG_RENDER_SHORT
 main.c, 282
FLAG_SATIN
 main.c, 282
FLAG_SATIN_SHORT
 main.c, 282
FLAG_SIERPINSKI_TRIANGLE
 main.c, 283
FLAG_SIMULATE
 main.c, 283
FLAG_STITCH
 main.c, 283
FLAG_STITCH_SHORT
 main.c, 283
FLAG_TEST
 main.c, 283
FLAG_TO
 main.c, 283
FLAG_TO_SHORT
 main.c, 284
FLAG_VERBOSE
 main.c, 284
FLAG_VERBOSE_SHORT
 main.c, 284
FLAG_VERSION
 main.c, 284
FLAG_VERSION_SHORT
 main.c, 284
flagList
 EmbPath_, 44
flags
 EmbStitch_, 52
formats.c
 binaryWriteInt, 257

binaryWriteIntBE, 257
 binaryWriteShort, 258
 binaryWriteUInt, 258
 binaryWriteUIntBE, 258
 binaryWriteUShort, 258
 binaryWriteUShortBE, 258
 emb_identify_format, 259
 embFormat_getExtension, 259
 embPattern_read, 259
 embPattern_readAuto, 259
 embPattern_write, 259
 embPattern_writeAuto, 260
 formatTable, 261
 fpad, 260
 fread_int16, 260
 fread_int32_be, 260
 fread_uint16, 260
 imageWithFrame, 261
 safe_free, 261
 formatTable
 embroidery.h, 146
 formats.c, 261
 fpad
 embroidery_internal.h, 196
 formats.c, 260
 fread_int16
 embroidery_internal.h, 196
 formats.c, 260
 fread_int32_be
 formats.c, 260
 fread_uint16
 embroidery_internal.h, 196
 formats.c, 260
 Fufu_Polyester
 embroidery.h, 108
 Fufu_Rayon
 embroidery.h, 108
 full_test_matrix
 embroidery.h, 142

 g
 EmbColor_, 30
 generate_dragon_curve
 fill.c, 243
 geometry
 EmbArray_, 26
 EmbLayer_, 40
 EmbPattern_, 45
 geometry.c
 embGeometry_boundingRect, 270
 embGeometry_free, 270
 embGeometry_init, 271
 embGeometry_move, 271
 embGeometry_vulcanize, 271
 get_trim_bounds
 main.c, 290
 getArcCenter
 embroidery.h, 143
 getArcDataFromBulge

 embroidery.h, 143
 getCircleCircleIntersections
 embroidery.h, 143
 getCircleTangentPoints
 embroidery.h, 143
 GetFile
 embroidery_internal.h, 196
 main.c, 291
 GREEN_TERM_COLOR
 embroidery_internal.h, 177
 haveExtraDIFATSectors
 main.c, 291
 header
 _bcf_file, 10
 height
 _vp3Hoop, 17
 EmblImage_, 38
 Hemingworth_Polyester
 embroidery.h, 109
 hex_code
 thread_color_, 64
 hilbert_curve
 embroidery.h, 143
 fill.c, 243
 hilbert_curve_l_system
 fill.c, 244
 home
 EmbPattern_, 45
 HOOP_110X110
 embroidery_internal.h, 178
 HOOP_126X110
 embroidery_internal.h, 178
 HOOP_140X200
 embroidery_internal.h, 178
 HOOP_230X200
 embroidery_internal.h, 178
 HOOP_50X50
 embroidery_internal.h, 178
 hoop_height
 EmbPattern_, 46
 hoop_width
 EmbPattern_, 46
 hoopSize
 ThredHeader_, 67
 hoopX
 ThredExtension_, 66
 hoopY
 ThredExtension_, 66
 hour
 EmbTime_, 56
 Huffman, 58
 default_value, 58
 lengths, 58
 nlengths, 59
 ntable, 59
 table, 59
 table_width, 59
 huffman

embroidery_internal.h, 186
huffman_build_table
compress.c, 81
embroidery_internal.h, 197
huffman_lookup
compress.c, 81
huffman_lookup_data
compress.c, 82
huffman_table_lookup
embroidery_internal.h, 197
hus_compress
compress.c, 81
embroidery_internal.h, 197
hus_decompress
compress.c, 81
embroidery_internal.h, 197
hus_thread
embroidery.h, 109
husThreads
embroidery.h, 146
thread-color.c, 341

image.c
image_diff, 274
writeImage, 274
image_diff
image.c, 274
imageWithFrame
embroidery_internal.h, 224
formats.c, 261
input_data
Compress, 22
input_length
Compress, 22
Isacord_Polyester
embroidery.h, 109
Isafil_Rayon
embroidery.h, 109

jef_thread
embroidery.h, 109
jefThreads
embroidery.h, 146
thread-color.c, 341
JUMP
embroidery.h, 109

L_system
embroidery.h, 119
layer
EmbPattern_, 46
left
_vp3Hoop, 17
EmbRect_, 49
left2
_vp3Hoop, 18
leftSiblingId
_bcf_directory_entry, 7
length
EmbArray_, 26
EmbSatinOutline_, 51
ThredHeader_, 67
lengths
Huffman, 58
LIBEMBROIDERY_EMBEDDED_VERSION
embroidery.h, 110
lindenmayer_system
embroidery.h, 144
fill.c, 244
line
EmbGeometry_, 35
LINETO
embroidery_internal.h, 178
lineType
EmbGeometry_, 35
EmbLine_, 42
EmbPath_, 44
EmbPoint_, 47
loadFatFromSector
embroidery_internal.h, 197
main.c, 291
LSYSTEM, 59
alphabet, 60
axiom, 60
constants, 60
rules, 60

Madeira_Polyester
embroidery.h, 110
Madeira_Rayon
embroidery.h, 110
magicCode
VipHeader_, 69
main.c
bcf_difat_create, 285
bcf_directory_free, 285
bcf_file_free, 285
bcfFile_read, 285
bcfFileFat_create, 285
bcfFileHeader_read, 286
binaryReadString, 286
binaryReadUnicodeString, 286
black_thread, 293
check_header_present, 286
CompoundFileDirectory, 286
CompoundFileDirectoryEntry, 287
copy_trim, 287
difatEntriesInHeader, 293
emb_error, 294
emb_optOut, 287
emb_readline, 287
emb_verbose, 294
embArc_print, 287
embColor_distance, 288
embColor_read, 288
embColor_write, 288
embConstantPi, 294
embSatinOutline_generateSatinOutline, 288

embSatinOutline_renderStitches, 288
 embThread_findNearestColor, 289
 embThread_findNearestThread, 289
 embThread_getRandom, 289
 embTime_initNow, 290
 embTime_time, 290
 embVector_print, 290
 entriesInDifatSector, 290
 FLAG_CIRCLE, 278
 FLAG_CIRCLE_SHORT, 278
 FLAG_COMBINE, 279
 FLAG_CROSS_STITCH, 279
 FLAG_ELLIPSE, 279
 FLAG_ELLIPSE_SHORT, 279
 FLAG_FILL, 279
 FLAG_FILL_SHORT, 279
 FLAG_FORMATS, 280
 FLAG_FORMATS_SHORT, 280
 FLAG_FULL_TEST_SUITE, 280
 FLAG_HELP, 280
 FLAG_HELP_SHORT, 280
 FLAG_HILBERT_CURVE, 280
 FLAG_LINE, 281
 FLAG_LINE_SHORT, 281
 FLAG_POLYGON, 281
 FLAG_POLYGON_SHORT, 281
 FLAG_POLYLINE, 281
 FLAG_POLYLINE_SHORT, 281
 FLAG QUIET, 282
 FLAG QUIET_SHORT, 282
 FLAG_RENDER, 282
 FLAG_RENDER_SHORT, 282
 FLAG_SATIN, 282
 FLAG_SATIN_SHORT, 282
 FLAG_SIERPINSKI_TRIANGLE, 283
 FLAG_SIMULATE, 283
 FLAG_STITCH, 283
 FLAG_STITCH_SHORT, 283
 FLAG_TEST, 283
 FLAG_TO, 283
 FLAG_TO_SHORT, 284
 FLAG_VERBOSE, 284
 FLAG_VERBOSE_SHORT, 284
 FLAG_VERSION, 284
 FLAG_VERSION_SHORT, 284
 get_trim_bounds, 290
 GetFile, 291
 haveExtraDIFATSectors, 291
 loadFatFromSector, 291
 NUM_FLAGS, 284
 parseDIFATSectors, 291
 parseDirectoryEntryName, 291
 parseTime, 292
 readFullSector, 292
 readNextSector, 292
 sectorSize, 292
 seekToSector, 292
 sizeOfChainingEntryAtEndOfDifatSector, 294
 sizeOfDifatEntry, 294
 sizeOfDirectoryEntry, 294
 sizeOfFatEntry, 295
 stringInArray, 293
 WHITESPACE, 295
 write_24bit, 293
 majorVersion
 _bcf_file_header, 13
 manufacturer_code
 thread_color_, 64
 Marathon_Polyester
 embroidery.h, 110
 Marathon_Rayon
 embroidery.h, 110
 MAX_STITCHES
 embroidery.h, 110
 MAX_THREADS
 embroidery.h, 111
 maxNumberOfDirectoryEntries
 _bcf_directory, 5
 Metro_Polyester
 embroidery.h, 111
 mid
 EmbArc_, 24
 miniSectorShift
 _bcf_file_header, 14
 miniStreamCutoffSize
 _bcf_file_header, 14
 minorVersion
 _bcf_file_header, 14
 minute
 EmbTime_, 56
 mitDecodeStitch
 embroidery_internal.h, 198
 encoding.c, 233
 mitEncodeStitch
 embroidery_internal.h, 198
 encoding.c, 233
 modifiedTime
 _bcf_directory_entry, 7
 modifierName
 ThredExtension_, 66
 month
 EmbTime_, 56
 MOVETO
 embroidery_internal.h, 179
 N_PES VERSIONS
 embroidery_internal.h, 179
 name
 EmblImage_, 38
 EmbLayer_, 40
 SvgAttribute_, 64
 thread_color_, 65
 negativeXHoopSize
 VipHeader_, 69
 negativeYHoopSize
 VipHeader_, 69
 next

_bcf_directory_entry, 7
nlenghts
 Huffman, 59
NORMAL
 embroidery.h, 111
ntable
 Huffman, 59
NUM_FLAGS
 main.c, 284
numberOfBytesRemaining
 _vp3Hoop, 18
numberOfColors
 _vp3Hoop, 18
 VipHeader_, 69
numberOfDifatSectors
 _bcf_file_header, 14
numberOfDirectorySectors
 _bcf_file_header, 14
numberOfEntriesInDifatSector
 embroidery_internal.h, 198
numberOfEntriesInFatSector
 _bcf_file_fat, 12
numberOfFATSectors
 _bcf_file_header, 14
numberOfFormats
 embroidery.h, 111
numberOfMiniFatSectors
 _bcf_file_header, 15
numberOfStitches
 VipHeader_, 69
numStiches
 ThredHeader_, 67

object
 EmbGeometry_, 35
objectType
 _bcf_directory_entry, 8
ObjectTypeRootEntry
 embroidery_internal.h, 179
ObjectTypeStorage
 embroidery_internal.h, 179
ObjectTypeStream
 embroidery_internal.h, 179
ObjectTypeUnknown
 embroidery_internal.h, 180

Pantone
 embroidery.h, 111
parseDIFATSectors
 main.c, 291
parseDirectoryEntryName
 main.c, 291
parseTime
 main.c, 292
path
 EmbGeometry_, 35
 EmblImage_, 38
pattern.c
 convert, 316

embPattern_addCircleAbs, 316
embPattern_addEllipseAbs, 316
embPattern_addLineAbs, 317
embPattern_addPathAbs, 317
embPattern_addPointAbs, 317
embPattern_addPolygonAbs, 317
embPattern_addPolylineObjectAbs, 318
embPattern_addRectAbs, 318
embPattern_addStitchAbs, 318
embPattern_addStitchRel, 318
embPattern_addThread, 319
embPattern_calcBoundingBox, 319
embPattern_center, 319
embPattern_changeColor, 319
embPattern_color_count, 319
embPattern_combineJumpStitches, 320
embPattern_copyPolylinesToStitch_list, 320
embPattern_copystitch_listToPolylines, 320
embPattern_correctForMaxStitchLength, 320
embPattern_create, 320
embPattern_designDetails, 321
embPattern_end, 321
embPattern_fixColorCount, 321
embPattern_flip, 321
embPattern_flipHorizontal, 321
embPattern_flipVertical, 322
embPattern_free, 322
embPattern_hideStitchesOverLength, 322
embPattern_jumpStitches, 322
embPattern_lengthHistogram, 322
embPattern_loadExternalColorFile, 323
embPattern_maximumStitchLength, 323
embPattern_minimumStitchLength, 323
embPattern_movePolylinesToStitch_list, 323
embPattern_movestitch_listToPolylines, 323
embPattern_realStitches, 324
embPattern_scale, 324
embPattern_totalStitchLength, 324
embPattern_trimStitches, 324

pcm_thread
 embroidery.h, 111
pcmThreads
 embroidery.h, 146
 thread-color.c, 341
pec_thread
 embroidery.h, 112
pecThreadCount
 embroidery.h, 146
 thread-color.c, 341
pecThreads
 embroidery.h, 147
 thread-color.c, 342
PES001
 embroidery_internal.h, 180
PES0020
 embroidery_internal.h, 180
PES0022
 embroidery_internal.h, 180

PES0030
 embroidery_internal.h, 180

PES0040
 embroidery_internal.h, 181

PES0050
 embroidery_internal.h, 181

PES0055
 embroidery_internal.h, 181

PES0056
 embroidery_internal.h, 181

PES0060
 embroidery_internal.h, 181

PES0070
 embroidery_internal.h, 181

PES0080
 embroidery_internal.h, 182

PES0090
 embroidery_internal.h, 182

PES0100
 embroidery_internal.h, 182

pfaffDecode
 embroidery_internal.h, 198
 encoding.c, 234

pfaffEncode
 embroidery_internal.h, 198
 encoding.c, 234

point
 EmbGeometry_, 36

pointList
 EmbPath_, 44

polygon
 EmbGeometry_, 36

polyline
 EmbGeometry_, 36

position
 EmbAlignedDim_, 22
 EmbAngularDim_, 23
 EmbArcLengthDim_, 25
 EmbBlock_, 28
 EmbDiameterDim_, 30
 EmblImage_, 38
 EmblInfiniteLine_, 39
 EmbLeaderDim_, 41
 EmbLinearDim_, 42
 EmbOrdinateDim_, 43
 EmbPoint_, 47
 EmbRadiusDim_, 48
 EmbRay_, 48
 EmbTextMulti_, 53
 EmbTextSingle_, 54

postitiveXHoopSize
 VipHeader_, 70

postitiveYHoopSize
 VipHeader_, 70

printArcResults
 embroidery_internal.h, 199

QUADTOCONTROL
 embroidery_internal.h, 182

QUADTOEND
 embroidery_internal.h, 182

r
 EmbColor_, 30

radians
 embroidery.h, 144

radius
 EmbCircle_, 29
 EmbEllipse_, 31
 EmbRect_, 49

read100
 embroidery_internal.h, 199

read10o
 embroidery_internal.h, 199

readArt
 embroidery_internal.h, 199

readBmc
 embroidery_internal.h, 199

readBro
 embroidery_internal.h, 200

readCnd
 embroidery_internal.h, 200

readCol
 embroidery_internal.h, 200

readCsd
 embroidery_internal.h, 200

readCsv
 embroidery_internal.h, 200

readDat
 embroidery_internal.h, 200

readDem
 embroidery_internal.h, 201

readDescriptions
 embroidery_internal.h, 201

readDsb
 embroidery_internal.h, 201

readDst
 embroidery_internal.h, 201

readDsz
 embroidery_internal.h, 201

readDxf
 embroidery_internal.h, 201

readEdr
 embroidery_internal.h, 202

readEmd
 embroidery_internal.h, 202

reader_state
 EmbFormatList_, 33

readExp
 embroidery_internal.h, 202

readExy
 embroidery_internal.h, 202

readEys
 embroidery_internal.h, 202

readFeatherPatterns
 embroidery_internal.h, 202

readFullSector
 embroidery_internal.h, 203

main.c, 292
readFxy
 embroidery_internal.h, 203
readGc
 embroidery_internal.h, 203
readGnc
 embroidery_internal.h, 203
readGt
 embroidery_internal.h, 203
readHoopName
 embroidery_internal.h, 203
readHus
 embroidery_internal.h, 204
readImageString
 embroidery_internal.h, 204
readInb
 embroidery_internal.h, 204
readInf
 embroidery_internal.h, 204
readJef
 embroidery_internal.h, 204
readKsm
 embroidery_internal.h, 204
readMax
 embroidery_internal.h, 205
readMit
 embroidery_internal.h, 205
readMotifPatterns
 embroidery_internal.h, 205
readNew
 embroidery_internal.h, 205
readNextSector
 embroidery_internal.h, 205
 main.c, 292
readOfm
 embroidery_internal.h, 205
readPcd
 embroidery_internal.h, 206
readPcm
 embroidery_internal.h, 206
readPcq
 embroidery_internal.h, 206
readPcs
 embroidery_internal.h, 206
readPec
 embroidery_internal.h, 206
readPecStitches
 embroidery_internal.h, 206
readPel
 embroidery_internal.h, 207
readPem
 embroidery_internal.h, 207
readPes
 embroidery_internal.h, 207
readPESHeaderV10
 embroidery_internal.h, 207
readPESHeaderV5
 embroidery_internal.h, 207
readPESHeaderV6
 embroidery_internal.h, 207
readPESHeaderV7
 embroidery_internal.h, 208
readPESHeaderV8
 embroidery_internal.h, 208
readPESHeaderV9
 embroidery_internal.h, 208
readPhb
 embroidery_internal.h, 208
readPhc
 embroidery_internal.h, 208
readPlt
 embroidery_internal.h, 208
readProgrammableFills
 embroidery_internal.h, 209
readRgb
 embroidery_internal.h, 209
readSew
 embroidery_internal.h, 209
readShv
 embroidery_internal.h, 209
readSst
 embroidery_internal.h, 209
readStx
 embroidery_internal.h, 209
readSvg
 embroidery_internal.h, 210
readT01
 embroidery_internal.h, 210
readT09
 embroidery_internal.h, 210
readTap
 embroidery_internal.h, 210
readThr
 embroidery_internal.h, 210
readThreads
 embroidery_internal.h, 210
readTxt
 embroidery_internal.h, 211
readU00
 embroidery_internal.h, 211
readU01
 embroidery_internal.h, 211
readVip
 embroidery_internal.h, 211
readVp3
 embroidery_internal.h, 211
readXxx
 embroidery_internal.h, 211
readZsk
 embroidery_internal.h, 212
rect
 EmbGeometry_, 36
RED_TERM_COLOR
 embroidery_internal.h, 182
report
 embroidery.h, 144

reserved
 ThredExtension_, 66
 ThredHeader_, 67
 reserved1
 _bcf_file_header, 15
 reserved2
 _bcf_file_header, 15
 RESET_TERM_COLOR
 embroidery_internal.h, 183
 reverse_byte_order
 encoding.c, 234
 right
 _vp3Hoop, 18
 EmbRect_, 50
 right2
 _vp3Hoop, 18
 rightSiblingId
 _bcf_directory_entry, 8
 RobisonAnton_Polyester
 embroidery.h, 112
 RobisonAnton_Rayon
 embroidery.h, 112
 rotation
 EmbEllipse_, 31
 EmbRect_, 50
 rules
 fill.c, 244
 LSYSTEM, 60
 safe_free
 embroidery_internal.h, 212
 formats.c, 261
 second
 EmbTime_, 57
 sectionName
 StxThread_, 61
 sectorShift
 _bcf_file_header, 15
 sectorSize
 _bcf_file_difat, 11
 main.c, 292
 seekToSector
 main.c, 292
 SEQUIN
 embroidery.h, 112
 shv_thread
 embroidery.h, 112
 shvThreadCount
 embroidery.h, 147
 thread-color.c, 342
 shvThreads
 embroidery.h, 147
 thread-color.c, 342
 side1
 EmbSatinOutline_, 51
 side2
 EmbSatinOutline_, 51
 Sigma_Polyester
 embroidery.h, 112
 signature
 _bcf_file_header, 15
 sigVersion
 ThredHeader_, 68
 sizeOfChainingEntryAtEndOfDifatSector
 main.c, 294
 sizeOfDifatEntry
 main.c, 294
 sizeOfDirectoryEntry
 main.c, 294
 sizeOfFatEntry
 main.c, 295
 someInt
 SubDescriptor_, 63
 someNum
 SubDescriptor_, 63
 someOtherInt
 SubDescriptor_, 63
 spline
 EmbGeometry_, 36
 src/array.c, 71, 75
 src/compress.c, 78, 82
 src/embroidery.h, 85, 148
 src/embroidery_internal.h, 155, 224
 src/encoding.c, 231, 235
 src/fill.c, 239, 245
 src/formats.c, 257, 262
 src/geometry.c, 270, 271
 src/image.c, 273, 274
 src/main.c, 276, 295
 src/pattern.c, 315, 325
 src/thread-color.c, 339, 342
 start
 EmbArc_, 24
 EmbBezier_, 27
 EmbLine_, 42
 startingSectorLocation
 _bcf_directory_entry, 8
 stateBits
 _bcf_directory_entry, 8
 stitch
 EmbArray_, 26
 EmbGeometry_, 36
 stitch_list
 EmbPattern_, 46
 stitchGranularity
 ThredExtension_, 66
 STOP
 embroidery.h, 113
 streamSize
 _bcf_directory_entry, 8
 streamSizeHigh
 _bcf_directory_entry, 8
 stringInArray
 embroidery_internal.h, 212
 main.c, 293
 stringVal
 VipHeader_, 70

stxColor
 StxThread_, 61

StxThread
 embroidery_internal.h, 186

StxThread_, 61
 colorCode, 61
 colorName, 61
 sectionName, 61
 stxColor, 61
 subDescriptors, 62

SubDescriptor
 embroidery_internal.h, 186

SubDescriptor_, 62
 colorCode, 62
 colorName, 62
 someInt, 63
 someNum, 63
 someOtherInt, 63

subDescriptors
 StxThread_, 62

Sulky_Rayon
 embroidery.h, 113

SVG_ATTRIBUTE
 embroidery_internal.h, 183

SVG_CATCH_ALL
 embroidery_internal.h, 183

SVG_Colors
 embroidery.h, 113

SVG_CREATOR_EMBROIDERMODDER
 embroidery_internal.h, 183

SVG_CREATOR_ILLUSTRATOR
 embroidery_internal.h, 183

SVG_CREATOR_INKSCAPE
 embroidery_internal.h, 183

SVG_CREATOR_NULL
 embroidery_internal.h, 184

SVG_ELEMENT
 embroidery_internal.h, 184

SVG_EXPECT_ATTRIBUTE
 embroidery_internal.h, 184

SVG_EXPECT_ELEMENT
 embroidery_internal.h, 184

SVG_EXPECT_NULL
 embroidery_internal.h, 184

SVG_EXPECT_VALUE
 embroidery_internal.h, 184

SVG_MEDIA_PROPERTY
 embroidery_internal.h, 185

SVG_NULL
 embroidery_internal.h, 185

SVG_PROPERTY
 embroidery_internal.h, 185

SvgAttribute
 embroidery_internal.h, 187

SvgAttribute_, 63
 name, 64
 value, 64

table
 Huffman, 59

table_width
 Huffman, 59

testEmbCircle
 embroidery_internal.h, 212

testEmbCircle_2
 embroidery_internal.h, 212

testEmbFormat
 embroidery_internal.h, 213

testGeomArc
 embroidery_internal.h, 213

testMain
 embroidery.h, 144

testTangentPoints
 embroidery_internal.h, 213

testThreadColor
 embroidery_internal.h, 213

text
 EmbTextMulti_, 53
 EmbTextSingle_, 54

thread
 EmbArray_, 26
 EmbGeometry_, 37

thread-color.c
 _dxfColorTable, 340
 brand_codes, 340
 brand_codes_files, 340
 husThreads, 341
 jefThreads, 341
 pcmThreads, 341
 pecThreadCount, 341
 pecThreads, 342
 shvThreadCount, 342
 shvThreads, 342
 threadColor, 339
 threadColorName, 340
 threadColorNum, 340

thread_color
 embroidery.h, 120

thread_color_, 64
 hex_code, 64
 manufacturer_code, 64
 name, 65

thread_list
 EmbPattern_, 46

ThreadArt_Polyester
 embroidery.h, 113

ThreadArt_Rayon
 embroidery.h, 113

threadColor
 embroidery.h, 144
 thread-color.c, 339

threadColorName
 embroidery.h, 144
 thread-color.c, 340

threadColorNum
 embroidery.h, 145
 thread-color.c, 340

ThreaDelight_Polyester
 embroidery.h, 113
 threadLength
 _vp3Hoop, 18
 ThredExtension
 embroidery_internal.h, 187
 ThredExtension_-, 65
 auxFormat, 65
 creatorName, 65
 hoopX, 66
 hoopY, 66
 modifierName, 66
 reserved, 66
 stitchGranularity, 66
 ThredHeader
 embroidery_internal.h, 187
 ThredHeader_-, 67
 hoopSize, 67
 length, 67
 numStiches, 67
 reserved, 67
 sigVersion, 68
 top
 _vp3Hoop, 19
 EmbRect_-, 50
 top2
 _vp3Hoop, 19
 transactionSignatureNumber
 _bcf_file_header, 15
 TRIM
 embroidery.h, 114
 type
 EmbArray_-, 26
 EmbFormatList_-, 33
 EmbGeometry_-, 37
 unknown
 VipHeader_-, 70
 unknown2
 _vp3Hoop, 19
 unknown3
 _vp3Hoop, 19
 unknown4
 _vp3Hoop, 19
 value
 SvgAttribute_-, 64
 vector
 EmbGeometry_-, 37
 vipDecodingTable
 embroidery.h, 147
 VipHeader
 embroidery_internal.h, 187
 VipHeader_-, 68
 attributeOffset, 68
 colorLength, 69
 magicCode, 69
 negativeXHoopSize, 69
 negativeYHoopSize, 69
 numberOfColors, 69
 numberOfStitches, 69
 postitiveXHoopSize, 70
 postitiveYHoopSize, 70
 stringVal, 70
 unknown, 70
 xOffset, 70
 yOffset, 70
 vp3Hoop
 embroidery_internal.h, 187
 WHITESPACE
 main.c, 295
 width
 _vp3Hoop, 19
 EmblImage_-, 39
 write100
 embroidery_internal.h, 213
 write10o
 embroidery_internal.h, 213
 write_24bit
 embroidery_internal.h, 214
 encoding.c, 234
 main.c, 293
 write_external_color_file
 EmbFormatList_-, 33
 writeArt
 embroidery_internal.h, 214
 writeBmc
 embroidery_internal.h, 214
 writeBro
 embroidery_internal.h, 214
 writeCnd
 embroidery_internal.h, 214
 writeCol
 embroidery_internal.h, 214
 writeCsd
 embroidery_internal.h, 215
 writeCsv
 embroidery_internal.h, 215
 writeDat
 embroidery_internal.h, 215
 writeDem
 embroidery_internal.h, 215
 writeDsb
 embroidery_internal.h, 215
 writeDst
 embroidery_internal.h, 215
 writeDsz
 embroidery_internal.h, 216
 writeDxf
 embroidery_internal.h, 216
 writeEdr
 embroidery_internal.h, 216
 writeEmd
 embroidery_internal.h, 216
 writeExp
 embroidery_internal.h, 216
 writeExy

embroidery_internal.h, 216
writeEys
 embroidery_internal.h, 217
writeFxy
 embroidery_internal.h, 217
writeGc
 embroidery_internal.h, 217
writeGnc
 embroidery_internal.h, 217
writeGt
 embroidery_internal.h, 217
writeHus
 embroidery_internal.h, 217
writeImage
 image.c, 274
writeInb
 embroidery_internal.h, 218
writeInf
 embroidery_internal.h, 218
writeJef
 embroidery_internal.h, 218
writeKsm
 embroidery_internal.h, 218
writeMax
 embroidery_internal.h, 218
writeMit
 embroidery_internal.h, 218
writeNew
 embroidery_internal.h, 219
writeOfm
 embroidery_internal.h, 219
writePcd
 embroidery_internal.h, 219
writePcm
 embroidery_internal.h, 219
writePcq
 embroidery_internal.h, 219
writePcs
 embroidery_internal.h, 219
writePec
 embroidery_internal.h, 220
writePecStitches
 embroidery_internal.h, 220
writePel
 embroidery_internal.h, 220
writePem
 embroidery_internal.h, 220
writePes
 embroidery_internal.h, 220
writePhb
 embroidery_internal.h, 220
writePhc
 embroidery_internal.h, 221
writePlt
 embroidery_internal.h, 221
writer_state
 EmbFormatList_, 33
writeRgb
 embroidery_internal.h, 221
writeSew
 embroidery_internal.h, 221
writeShv
 embroidery_internal.h, 221
writeSst
 embroidery_internal.h, 221
writeStx
 embroidery_internal.h, 222
writeSvg
 embroidery_internal.h, 222
writeT01
 embroidery_internal.h, 222
writeT09
 embroidery_internal.h, 222
writeTap
 embroidery_internal.h, 222
writeThr
 embroidery_internal.h, 222
writeTxt
 embroidery_internal.h, 223
writeU00
 embroidery_internal.h, 223
writeU01
 embroidery_internal.h, 223
writeVip
 embroidery_internal.h, 223
writeVp3
 embroidery_internal.h, 223
writeXxx
 embroidery_internal.h, 223
writeZsk
 embroidery_internal.h, 224

x
 EmbStitch_, 52
 EmbVector_, 57
xOffset
 _vp3Hoop, 20
 VipHeader_, 70

y
 EmbStitch_, 53
 EmbVector_, 58
year
 EmbTime_, 57
YELLOW_TERM_COLOR
 embroidery_internal.h, 185
yOffset
 _vp3Hoop, 20
 VipHeader_, 70

Z102_Isacord_Polyester
 embroidery.h, 114