scmsg

COLLABORATORS

	TITLE : scmsg	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 scmsg **1**

1.1 SAS/C Error/Warning Messages . . . 1
1.2 compiler . . . 1
1.3 linker . . . 6
1.4 cplusplus . . . 7
1.5 strict . . . 14
1.6 ANSI COMPILER FLAG . . . 14
1.7 scmxx1 . . . 14
1.8 scmxx2 . . . 15
1.9 scmxx3 . . . 15
1.10 scmxx4 . . . 15
1.11 scmxx5 . . . 15
1.12 scmxx6 . . . 15
1.13 scmxx7 . . . 16
1.14 scmxx8 . . . 16
1.15 scmxx9 . . . 16
1.16 scmx10 . . . 16
1.17 scmx11 . . . 16
1.18 scmx12 . . . 17
1.19 scmx13 . . . 17
1.20 scm001 . . . 17
1.21 scm002 . . . 17
1.22 scm003 . . . 17
1.23 scm004 . . . 18
1.24 scm005 . . . 18
1.25 scm006 . . . 18
1.26 scm007 . . . 18
1.27 scm008 . . . 19
1.28 scm009 . . . 19
1.29 scm010 . . . 19

1.30 scm011 . . . 19
1.31 scm012 . . . 20
1.32 scm013 . . . 20
1.33 scm014 . . . 20
1.34 scm015 . . . 20
1.35 scm016 . . . 21
1.36 scm017 . . . 21
1.37 scm018 . . . 21
1.38 scm019 . . . 21
1.39 scm020 . . . 22
1.40 scm021 . . . 22
1.41 scm022 . . . 22
1.42 scm023 . . . 22
1.43 scm024 . . . 23
1.44 scm025 . . . 23
1.45 scm026 . . . 24
1.46 scm027 . . . 24
1.47 scm028 . . . 24
1.48 scm029 . . . 24
1.49 scm030 . . . 25
1.50 scm031 . . . 25
1.51 scm032 . . . 25
1.52 scm033 . . . 25
1.53 scm034 . . . 26
1.54 scm035 . . . 26
1.55 scm036 . . . 26
1.56 scm037 . . . 27
1.57 scm038 . . . 27
1.58 scm039 . . . 27
1.59 scm040 . . . 27
1.60 scm041 . . . 27
1.61 scm042 . . . 28
1.62 scm043 . . . 28
1.63 scm044 . . . 28
1.64 scm045 . . . 29
1.65 scm046 . . . 29
1.66 scm047 . . . 29
1.67 scm048 . . . 29
1.68 scm049 . . . 29

1.69 scm050 . . . 30
1.70 scm051 . . . 30
1.71 scm052 . . . 30
1.72 scm053 . . . 30
1.73 scm054 . . . 30
1.74 scm055 . . . 31
1.75 scm056 . . . 31
1.76 scm057 . . . 31
1.77 scm058 . . . 31
1.78 scm059 . . . 31
1.79 scm060 . . . 32
1.80 scm061 . . . 32
1.81 scm062 . . . 32
1.82 scm063 . . . 33
1.83 scm064 . . . 33
1.84 scm065 . . . 33
1.85 scm066 . . . 33
1.86 scm067 . . . 34
1.87 scm068 . . . 34
1.88 scm069 . . . 34
1.89 scm070 . . . 34
1.90 scm071 . . . 35
1.91 scm072 . . . 35
1.92 scm073 . . . 36
1.93 scm074 . . . 36
1.94 scm075 . . . 36
1.95 scm076 . . . 37
1.96 scm077 . . . 37
1.97 scm078 . . . 37
1.98 scm079 . . . 37
1.99 scm080 . . . 38
1.100 scm081 . . . 38
1.101 scm082 . . . 38
1.102 scm083 . . . 39
1.103 scm084 . . . 39
1.104 scm085 . . . 39
1.105 scm086 . . . 40
1.106 scm087 . . . 40
1.107 scm088 . . . 40

1.108 scm089 . . . 40
1.109 scm090 . . . 41
1.110 scm091 . . . 41
1.111 scm092 . . . 41
1.112 scm093 . . . 42
1.113 scm094 . . . 42
1.114 scm095 . . . 43
1.115 scm096 . . . 43
1.116 scm097 . . . 43
1.117 scm098 . . . 43
1.118 scm099 . . . 44
1.119 scm100 . . . 44
1.120 scm101 . . . 44
1.121 scm102 . . . 44
1.122 scm103 . . . 45
1.123 scm104 . . . 45
1.124 scm105 . . . 46
1.125 scm106 . . . 46
1.126 scm107 . . . 46
1.127 scm108 . . . 46
1.128 scm109 . . . 47
1.129 scm110 . . . 47
1.130 scm111 . . . 47
1.131 scm112 . . . 47
1.132 scm113 . . . 48
1.133 scm114 . . . 48
1.134 scm115 . . . 48
1.135 scm116 . . . 49
1.136 scm117 . . . 49
1.137 scm118 . . . 49
1.138 scm119 . . . 49
1.139 scm120 . . . 50
1.140 scm121 . . . 50
1.141 scm122 . . . 51
1.142 scm123 . . . 51
1.143 scm125 . . . 51
1.144 scm126 . . . 52
1.145 scm127 . . . 52
1.146 scm128 . . . 52

1.147scm129 52
1.148scm131 53
1.149scm132 53
1.150scm133 54
1.151scm134 54
1.152scm135 54
1.153scm136 54
1.154scm137 55
1.155scm138 55
1.156scm139 55
1.157scm140 55
1.158scm142 56
1.159scm143 56
1.160scm146 56
1.161scm147 57
1.162scm148 57
1.163scm149 57
1.164scm150 58
1.165scm152 58
1.166scm154 59
1.167scm155 59
1.168scm156 59
1.169scm158 60
1.170scm159 60
1.171scm161 60
1.172scm162 60
1.173scm163 60
1.174scm164 61
1.175scm165 61
1.176scm166 61
1.177scm169 61
1.178scm170 62
1.179scm176 62
1.180scm178 63
1.181scm179 63
1.182scm180 63
1.183scm181 63
1.184scm182 64
1.185scm183 64

1.186scm184 64
1.187scm185 64
1.188scm186 64
1.189scm187 65
1.190scm188 65
1.191scm189 66
1.192scm190 66
1.193scm192 66
1.194scm193 67
1.195scm194 67
1.196scm195 68
1.197scm196 68
1.198scm198 68
1.199scm199 68
1.200scm200 68
1.201scm202 69
1.202scm204 69
1.203scm209 69
1.204scm212 70
1.205scm213 70
1.206scm216 70
1.207scm217 71
1.208scm218 71
1.209scm220 71
1.210scm223 71
1.211scm224 72
1.212scm225 72
1.213scm226 73
1.214scm301 73
1.215scm302 73
1.216scm303 73
1.217scm304 73
1.218scm305 74
1.219scm306 74
1.220scm307 74
1.221scm308 74
1.222scm402 75
1.223scm403 75
1.224scm404 75

1.225 scm405 . . . 75
1.226 scm406 . . . 75
1.227 scm407 . . . 76
1.228 scm408 . . . 76
1.229 scm409 . . . 76
1.230 scm410 . . . 76
1.231 scm411 . . . 76
1.232 scm412 . . . 76
1.233 scm415 . . . 77
1.234 scm416 . . . 77
1.235 scm417 . . . 77
1.236 scm1101 . . . 77
1.237 scm1102 . . . 77
1.238 scm1103 . . . 77
1.239 scm1104 . . . 78
1.240 scm1105 . . . 78
1.241 scm1106 . . . 78
1.242 scm1107 . . . 78
1.243 scm1108 . . . 78
1.244 scm1109 . . . 79
1.245 scm1110 . . . 79
1.246 scm1111 . . . 79
1.247 scm1112 . . . 79
1.248 scm1113 . . . 79
1.249 scm1114 . . . 79
1.250 scm1115 . . . 80
1.251 scm1116 . . . 80
1.252 scm1117 . . . 80
1.253 scm1118 . . . 80
1.254 scm1119 . . . 80
1.255 scm1121 . . . 80
1.256 scm1122 . . . 81
1.257 scm1123 . . . 81
1.258 scm1124 . . . 81
1.259 scm1125 . . . 81
1.260 scm1126 . . . 81
1.261 scm1127 . . . 81
1.262 scm1129 . . . 82
1.263 scm1130 . . . 82

1.264scm1200 82
1.265scm1205 82
1.266scm1206 82
1.267scm1208 83
1.268scm1319 83
1.269scm1320 83
1.270scm1321 83
1.271scm1322 83
1.272scm1323 83
1.273scm1324 84
1.274scm1325 84
1.275scm1326 84
1.276scm1327 84
1.277scm1328 84
1.278scm1329 84
1.279scm1330 85
1.280scm1331 85
1.281scm1332 85
1.282scm1334 85
1.283scm1335 86
1.284scm1336 86
1.285scm1337 86
1.286scm1338 86
1.287scm1339 86
1.288scm1340 87
1.289scm1342 87
1.290scm1343 87
1.291scm1344 87
1.292scm1345 87
1.293scm1346 87
1.294scm1347 88
1.295scm1348 88
1.296scm1349 88
1.297scm1350 88
1.298scm1351 88
1.299scm1352 89
1.300scm1353 89
1.301scm1354 89
1.302scm1355 89

1.303scm1356 89
1.304scm1357 90
1.305scm1358 90
1.306scm1359 90
1.307scm1361 90
1.308scm1362 90
1.309scm1363 90
1.310scm1364 91
1.311scm1365 91
1.312scm1366 91
1.313scm1367 91
1.314scm1368 91
1.315scm1369 91
1.316scm1370 92
1.317scm1371 92
1.318scm1372 92
1.319scm1373 92
1.320scm1374 92
1.321scm1375 92
1.322scm1376 93
1.323scm1377 93
1.324scm1378 93
1.325scm1379 93
1.326scm1380 93
1.327scm1382 93
1.328scm1383 94
1.329scm1384 94
1.330scm1385 94
1.331scm1386 94
1.332scm1387 94
1.333scm1388 94
1.334scm1389 95
1.335scm1391 95
1.336scm1392 95
1.337scm1393 95
1.338scm1394 95
1.339scm1395 96
1.340scm1396 96
1.341scm1397 96

1.342scm1398 96
1.343scm1399 97
1.344scm1400 97
1.345scm1401 97
1.346scm1402 98
1.347scm1403 98
1.348scm1404 98
1.349scm1406 98
1.350scm1407 98
1.351scm1408 99
1.352scm1409 99
1.353scm1410 99
1.354scm1411 99
1.355scm1412 99
1.356scm1413 99
1.357scm1414 99
1.358scm1415 100
1.359scm1416 100
1.360scm1417 100
1.361scm1418 100
1.362scm1419 100
1.363scm1420 100
1.364scm1421 100
1.365scm1422 101
1.366scm1423 101
1.367scm1424 101
1.368scm1425 101
1.369scm1426 101
1.370scm1427 101
1.371scm1428 102
1.372scm1429 102
1.373scm1430 102
1.374scm1431 102
1.375scm1432 102
1.376scm1433 103
1.377scm1434 103
1.378scm1435 103
1.379scm1436 103
1.380scm1437 103

1.381scm1438 . . . 103
1.382scm1439 . . . 104
1.383scm1440 . . . 104
1.384scm1441 . . . 104
1.385scm1442 . . . 104
1.386scm1443 . . . 104
1.387scm1444 . . . 104
1.388scm1445 . . . 105
1.389scm1446 . . . 105
1.390scm1447 . . . 105
1.391scm1448 . . . 105
1.392scm1449 . . . 105
1.393scm1450 . . . 105
1.394scm1451 . . . 106
1.395scm1452 . . . 106
1.396scm1453 . . . 106
1.397scm1454 . . . 106
1.398scm1455 . . . 106
1.399scm1456 . . . 106
1.400scm1457 . . . 107
1.401scm1458 . . . 107
1.402scm1460 . . . 107
1.403scm1461 . . . 107
1.404scm1462 . . . 107
1.405scm1463 . . . 108
1.406scm1464 . . . 108
1.407scm1465 . . . 108
1.408scm1467 . . . 108
1.409scm1468 . . . 108
1.410scm1469 . . . 108
1.411scm1472 . . . 109
1.412scm1473 . . . 109
1.413scm1474 . . . 109
1.414scm1475 . . . 109
1.415scm1476 . . . 109
1.416scm1477 . . . 109
1.417scm1478 . . . 110
1.418scm1479 . . . 110
1.419scm1480 . . . 110

1.420scm1481 110
1.421scm1482 111
1.422scm1483 111
1.423scm1484 111
1.424scm1485 111
1.425scm1486 111
1.426scm1487 111
1.427scm1489 112
1.428scm1490 112
1.429scm1491 112
1.430scm1492 112
1.431scm1493 112
1.432scm1494 112
1.433scm1495 113
1.434scm1498 113
1.435scm1499 113
1.436scm1500 113
1.437scm1501 113
1.438scm1502 113
1.439scm1503 114
1.440scm1504 114
1.441scm1506 114
1.442scm1507 114
1.443scm1508 114
1.444scm1509 114
1.445scm1510 114
1.446scm1511 115
1.447scm1512 115
1.448scm1513 115
1.449scm1514 115
1.450scm1515 115
1.451scm1516 116
1.452scm1517 116
1.453scm1518 116
1.454scm1522 116
1.455scm1523 116
1.456scm1524 116
1.457scm1525 117
1.458scm1528 117

1.459scm1530 . . . 117
1.460scm1531 . . . 117
1.461scm1532 . . . 117
1.462scm1533 . . . 117
1.463scm1534 . . . 118
1.464scm1535 . . . 118
1.465scm1536 . . . 118
1.466scm1537 . . . 118
1.467scm1538 . . . 118
1.468scm1539 . . . 118
1.469scm1540 . . . 119
1.470scm1541 . . . 119
1.471scm1542 . . . 119
1.472scm1543 . . . 119
1.473scm1544 . . . 119
1.474scm1545 . . . 120
1.475scm1546 . . . 120
1.476scm1547 . . . 120
1.477scm1548 . . . 120
1.478scm1549 . . . 120
1.479scm1550 . . . 120
1.480scm1551 . . . 121
1.481scm1553 . . . 121
1.482scm1554 . . . 121
1.483scm1555 . . . 121
1.484scm1556 . . . 121
1.485scm1557 . . . 121
1.486scm1558 . . . 122
1.487scm1559 . . . 122
1.488scm1560 . . . 122
1.489scm1562 . . . 122
1.490scm1564 . . . 122
1.491scm1565 . . . 122
1.492scm1566 . . . 123
1.493scm1567 . . . 123
1.494scm1568 . . . 123
1.495scm1569 . . . 123
1.496scm1570 . . . 123
1.497scm1571 . . . 124

1.498 scm1572 . . . 124
1.499 scm1573 . . . 124
1.500 scm1574 . . . 124
1.501 scm1575 . . . 124
1.502 scm1576 . . . 124
1.503 scm1577 . . . 125
1.504 scm1578 . . . 125
1.505 scm1579 . . . 125
1.506 scm1580 . . . 125
1.507 scm1581 . . . 126
1.508 scm1582 . . . 126
1.509 scm1583 . . . 126
1.510 scm1584 . . . 126
1.511 scm1585 . . . 126
1.512 scm1586 . . . 126
1.513 scm1587 . . . 127
1.514 scm1588 . . . 127
1.515 scm1589 . . . 127
1.516 scm1590 . . . 127
1.517 scm1591 . . . 127
1.518 scm1592 . . . 127
1.519 scm1593 . . . 128
1.520 scm1594 . . . 128
1.521 scm1597 . . . 128
1.522 scm1610 . . . 128
1.523 scm1611 . . . 129
1.524 scm1612 . . . 129
1.525 scm1613 . . . 129
1.526 scm1614 . . . 129
1.527 scm1615 . . . 129
1.528 scm1616 . . . 130
1.529 slm103 . . . 130
1.530 slm425 . . . 130
1.531 slm426 . . . 130
1.532 slm443 . . . 130
1.533 slm444 . . . 130
1.534 slm445 . . . 131
1.535 slm446 . . . 131
1.536 slm447 . . . 131

1.537slm448 . . . 131
1.538slm449 . . . 131
1.539slm450 . . . 131
1.540slm501 . . . 132
1.541slm502 . . . 132
1.542slm503 . . . 132
1.543slm504 . . . 132
1.544slm505 . . . 133
1.545slm506 . . . 133
1.546slm507 . . . 133
1.547slm508 . . . 133
1.548slm509 . . . 133
1.549slm510 . . . 134
1.550slm512 . . . 134
1.551slm513 . . . 134
1.552slm514 . . . 134
1.553slm515 . . . 135
1.554slm516 . . . 135
1.555slm600 . . . 135
1.556slm601 . . . 135
1.557slm602 . . . 135
1.558slm603 . . . 136
1.559slm604 . . . 136
1.560slm605 . . . 136
1.561slm607 . . . 136
1.562slm608 . . . 136
1.563slm609 . . . 136
1.564slm610 . . . 136
1.565slm611 . . . 137
1.566slm612 . . . 137
1.567slm613 . . . 137
1.568slm614 . . . 137
1.569slm615 . . . 137
1.570slm616 . . . 138
1.571slm617 . . . 138
1.572slm618 . . . 138
1.573slm619 . . . 138
1.574slm620 . . . 138
1.575slm621 . . . 138

1.576 slm622 139
1.577 slm623 139
1.578 slm624 139
1.579 slm625 139
1.580 slm626 139
1.581 slm627 140
1.582 HELP 140

Chapter 1

scmsg

1.1 SAS/C Error/Warning Messages

```
        Effect of Specific Compiler Flags on Compiler Error Messages
                    STRICT                  ANSI


                  Compiler Error and Warning Messages

                    C++ Error and Warning Messages

                         Linker Error Messages

                       Common Problems Compiling

                        Common Problems Linking
```

1.2 compiler

```
    ***       Can't delete old GST:  object is in use
               Continuing with no GST file
    ***       Can't open GST file:  <gst-filename>
    ***       Can't open sc:libs/<lib-name>.library
    ***       Can't open <type> file "<name>" for <mode>
    ***       Combined output filename too long
    ***       CXERR:  <num>
    ***       CXWRN:  <text>
    ***       Floating point overflow optimizing constants
    ***       Freeing Resources
    ***       Invalid symbol definition:  <symbol-name>
    ***       I/O error code on file "<name>"
    ***       Seek error on object file
    ***       Warning:  Debugging information may be incorrect for
               optimized code.

Warning   1   invalid preprocessor command
Error     2   unexpected end of file
Error     3   file not found "<filename>"
```

```
Error     4   invalid lexical token
Error     5   invalid macro usage
Error     6   line buffer overflow
Warning   7   register parameters require a prototype
                Stack parameters used
Error     8   invalid conversion
Error     9   undefined identifier "<name>"
Error    10   invalid subscript expression
Error    11   string not terminated
Error    12   invalid structure reference
Error    13   member name missing
Error    14   undefined member "<name>"
Error    15   invalid function call
Error    16   invalid function argument
Error    17   too many operands
Warning  18   non-ANSI use of operator in preprocessor condition
Error    19   unbalanced parentheses
Error    20   invalid constant expression
Error    21   illegal use of struct, union, or array type
Error    22   __asm functions cannot accept structure or union arguments
                Use pointers instead
Error    23   invalid use of conditional operator
Error    24   pointer operand required
Error    25   modifiable lvalue required
Error    26   arithmetic operand required
Error    27   arithmetic or pointer operand required
Error    28   missing operand
Error    29   operation cannot be performed on a pointer
Warning  30   pointers do not point to same type of object
Error    31   integral operand required
Error    32   cannot convert to required type
Warning  33   non-portable operation on structure or union
Error    34   invalid initializer expression
Error    35   closing brace expected
Warning  36   control cannot reach this statement
Error    37   duplicate statement label "<name>"
                See line <number> in file "<filename>"
Error    38   unbalanced braces
Error    39   invalid use of keyword "<keyword>"
Error    40   break not inside loop or switch
Error    41   case not inside switch
Warning  42   case expression not integral
Error    43   duplicate of case value
                See line <number> in file "<filename>"
Error    44   continue not inside loop
Error    45   default not inside switch
Error    46   duplicate default
                See line <number> in file "<filename>"
Error    47   while missing from do statement
Error    48   invalid while expression
Error    49   else not associated with if
Error    50   label missing from goto
Warning  51   C++ comment detected
Error    52   invalid if expression
Error    53   invalid return expression
Warning  54   switch expression not integral
Warning  55   no case values for switch statement
```

```
Error    56   colon expected
Error    57   semi-colon expected
Error    58   missing parenthesis
Error    59   invalid storage class
Error    60   incompatible struct, union or array types
Error    61   undefined struct/union tag "<tag-name>"
Warning  62   constant <number> out of range for type "type"
               Valid range is <low> to <high>
Warning  63   item "<name>" already declared
               See line <number> file "<filename>"
Error    64   structure contains no members
Error    65   invalid function definition
Warning  66   invalid array limit expression
Error    67   illegal object
Error    68   illegal object for structure
Error    69   struct <name> includes instance of self
Warning  70   unrecognized escape sequence
Error    71   formal declaration error "<name>"
Error    72   conflict with previous declaration
               See line <number> file "<filename>"
Warning  73   declaration expected
Warning  74   initializer data truncated
Error    75   invalid sizeof expression
Error    76   left brace expected
Error    77   identifier expected
Error    78   undefined statement label "<name>"
Warning  79   duplicate of enumeration value
               See line <line> file "<filename>"
Warning  80   invalid bit field or misplaced ':'
Error    81   preprocessor symbol loop; macro expansion too long or
               circular
Error    82   maximum object/storage size exceeded
               Size limit for this class is <number>
Warning  83   reference beyond object size
Warning  84   redefinition of pragma or preprocessor symbol "<name>"
               See line <number> file "<filename>"
Warning  85   return value mismatch for function "<name>"
               Expecting "<type1>", found "<type2>"
Warning  86   formal parameters conflict with prototype
               See line <number> file "<filename>"
Warning  87   argument count incorrect, expecting <number> arguments
               See line <number> file "<filename>"
Warning  88   argument type incorrect
               Expecting "<type1>", found "<type2>"
Warning  89   constant converted from "<type1>" to "<type2>"
Error    90   invalid argument type specifier
Error    91   illegal void operand
Warning  92   statement has no effect
Warning  93   no reference to identifier "<name>"
Warning  94   uninitialized auto variable "<name>"
Warning  95   unrecognized #pragma operand
Error    96   missing name for #pragma
Error    97   bad library base for #pragma
Error    98   invalid data for #pragma
Error    99   attempt to change a const lvalue
Warning 100   no prototype declared for function "<name>"
Error   101   redundant keywords in declaration
```

```
Error   102   conflicting keywords in declaration
Warning 103   uninitialized constant "[name]"
Warning 104   conversion from pointer to const/volatile to pointer to
               non-const/volatile
Warning 105   module does not define any externally-known symbols
Error   106   postfix expression not allowed on a constant
Error   107   too many initializers
Warning 108   zero-length arrays are not an ANSI feature
Error   109   invalid use of type name or keyword
Warning 110   enum constant expression is wrong type
               Expecting "<type1>", found "<type2>"
Warning 111   non-portable enum type specified
Warning 112   include file "filename" not in GST
Warning 113   invalid structure reference
               "<op>" operator invalid for type "<type>"
Warning 114   negative shift or shift too big for type shifts for type
               "<type>" must be between 0 and <number> bits
Error   115   enum constant value "<number>" out of range for enum type
Warning 116   undefined enum tag "<name>"
Error   117   enum contains no members
Error   118   conflicting use of enum/struct/union tag "<name>"
Error   119   identifiers missing from definition of function "<name>"
Warning 120   Integral type mismatch:  possible portability problem
               Expecting "<type1>", found "<type2>"
Warning 121   hex/octal constant "<constant>" too large for char
               High bits may be lost
Warning 122   missing ellipsis
Warning 123   no tag defined for enumeration
               Cannot construct prototype
Error   125   invalid number
Warning 126   #endif, #else, or #elif out of order
Error   127   operand to # operator must be a macro argument
Error   128   text-from-#error
Error   129   ambiguous struct or union member "<name>"
Error   131   maximum temporary or formal storage exceeded
Warning 132   extra tokens after valid preprocessor directive
Error   133   cannot redefine macro "<name>"
Warning 134   too many arguments
Error   135   argument count incorrect for macro "<name>", expecting
               <number> arguments
               See line <number> file "<filename>"
Error   136   invalid use of register keyword
Warning 137   ANSI limits #line numbers to between 1 and 32767
Error   138   operation invalid for pointer to void
Warning 139   missing #endif
               See line <number> file "<filename>"
Warning 140   sizeof operator used on array that has been converted to
               pointer
Error   142   array size never given for "<name>"
Error   143   object has no address
Warning 146   case value out of range for switch type
Warning 147   conversion between function and data pointers
Warning 148   use of incomplete struct/union/enum tag "<name>"
               See line <number> file "<filename>"
Warning 149   incomplete struct/union/enum tag in prototype scope "<name>"
Warning 150   the keyword "<name>" is meaningless for <itemtype>
Error   152   cannot define function via typedef name
```

```
Warning 154   no prototype declared for function pointer
Warning 155   no statement after label
Warning 156   operation/comparison of pointer to "int" and pointer to
               "type"
Error   158   invalid type name
Warning 159   use of unary minus on unsigned value
Warning 161   no prototype declared at definition for function "<name>"
Warning 162   non-ANSI use of ellipsis punctuator
Warning 163   initialization of auto struct, union, or array
Warning 164   & applied to array
Warning 165   use of narrow type in prototype
Error   166   unrecoverable error or too many errors
               Terminating compilation
Warning 169   incompatible operands of conditional operator (?:)
               "<type1>" conflicts with "<type2>"
Warning 170   overflow during operation on constants
Warning 176   implicitly promoted formal "<name>" conflicts with prototype
               See line <number> file "<filename>"
Warning 178   indirect call without indirection operator
Warning 179   narrow type used in old-style definition
Warning 180   no space between macro name and its replacement list
Warning 181   "<name>" was declared both static and external
               See line <number> file "<filename>"
Warning 182   static function "<name>" declared but not defined
               See line <number> file "<filename>"
Warning 183   inline function declared but not defined
               See line <number> file "<filename>"
Warning 184   unterminated character constant
Error   185   comma expected
Warning 186   implicit conversion between pointer and scalar
Warning 187   negative value assigned to unsigned type
Error   188   Function and data definitions not allowed when creating a GST
Warning 189   <option> option differs from the one used to build the GST
Warning 190   #include ignored because header already included
              See line <number> file "<filename>"
Warning 192   wrong size for enum
Warning 193   implicit reference to struct/union member
               Reference assumed to be "<reference>"
Warning 194   too much local data for NEAR reference, some changed to
               FAR
Warning 195   nested comment detected
Warning 196   specified include directory not found:  "<name>"
Warning 198   __regargs and __asm cannot be used on a varargs function
Error   199   unbalanced comment
               See line <number> file "<filename>"
Error   200   no register specified for parameter to __asm function
Warning 202   relational comparison between unsigned and zero
Warning 204   macro invocation not terminated
Warning 209   macro invocation may have multiple side effects
Warning 212   item "<name>" already declared
               See line <number> file "<filename>"
Warning 213   empty argument to preprocessor macro
Warning 216   symbol "<name>" found
Warning 217   macro invocation may call function multiple times
Error   218   declaration found in statement block
Warning 220   old-fashioned assignment operator taken as "<operators>"
Error   223   "<filename>" is not a valid GST file
```

```
Warning 224   item "<name>" already defined
               See line <number> file "<filename>"
Warning 225   pointer type mismatch
               "<type1>" does not match "<type2>"
Error   226   cannot convert "type1" to "type2"
Warning 301   Indirect reference through NULL pointer.
Warning 302   Type punning involves representation change <symbol>
Warning 303   Reference has overlapped definition <symbol>
Warning 304   Dead assignment eliminated <symbol>
Warning 305   Uninitialized variable <symbol>
Note    306   <reason> function inlined:  <function name> {from
               line <u>}
Warning 307   Return value missing in inline function
Note    308   Inline function does not use formal parameter <symbol>
Error   402   Wrong number of parms to builtin function
Error   403   Argument(s) to function-name must be int type
Error   404   __builtin_fpc requires MATH=68881 option
Error   405   Floating point opcode must be a constant.
Error   406   Offset from library base must be a constant
Error   407   Offset from library base must be negative
Error   408   Insufficient parameters for library call
Error   409   Too many parameters for library call
Error   410   Invalid register specification for getreg/putreg
Error   411   Value for getreg/putreg must be an integral type
Error   412   FP register used without co-processor
Error   415   Same register used twice for parameters
Error   416   No register specified for ASM call
Error   417   No Data register available to reach far formals.
               Reduce the size of auto variables or reduce the
               number of register parameters.
```

1.3 linker

```
Error 103   Out of memory!!
Error 425   Cannot find library <library-name>
Error 426   Cannot find object name
Error 443   <filename> is an invalid file name
Error 444   Hunk_Symbol has bad <symbol-type> symbol <symbol-name>
Error 445   Invalid HUNK_SYMBOL <symbol-name>
Error 446   Invalid symbol type <symbol-type> for <symbol-name>
Error 447   <filename> is a load file
Error 448   <filename> is not a valid object file
Error 449   No hunk_end seen for <filename>
Error 450   Object file <filename> is an extended library
Error 501   Invalid Reloc 8 or 16 reference
Error 502   <function-name> symbol - Distance for Reloc 16 greater than
            32768
Error 503   <function-name> symbol - Distance for Reloc 8 greater than 128
Error 504   <variable-name> symbol - Distance for Data Reloc 16 greater
            than 32768
Error 505   <variable-name> symbol - Distance for Data Reloc 8 greater than
            128
Error 506   Can't locate resolved symbol <symbol-name>
Error 507   Unknown Symbol type <symbol-type>, for symbol <symbol-name>
Error 508   Symbol type <symbol-type> unimplemented
```

```
Error 509   Unknown hunk type <symbol-type> in Pass2
Error 510   <symbol-name> symbol - Near reference to a data item not in
            near data section
Error 512   Invalid branch to <function-name> in overlay Node <module-name>
Error 513   Multiple NTRYHUNK segments not permitted
Error 514   Overlay manager _ovlyMgr is undefined
Error 515   An ALV was generated pointing to data <variable-name> symbol
Error 516   Attempt to merge BSS with CODE or CODE/DATA
Error 600   Invalid command <command>
Error 601   <option> option specified more than once
Error 602   Unable to open output file <filename>
Error 603   <string> is not a valid number
Error 604   with file is not readable
Error 605   Cannot open with file <filename>
Error 607   No FROM/ROOT files specified
Error 608   Premature EOF encountered
Error 609   Error seeking in file <filename>
Error 610   <module-name> has no parent in overlay tree
Error 611   Reloc found with odd address for symbol <symbol-name>, file
            <filename>
Error 612   MERGED Data relocation to non-code section in Overlay Node
            Reference at offset <hex-address> in <module-name>,
            To Unit <module-name>
Error 613   MERGED Data relocation to static function is not resolvable
            by Overlay Manager.
            Reference at <offset> hex-address in <filename>, To
            Unit <filename>
Error 614   More than one MERGED data section found
Error 615   Code hunk named __MERGED
Error 616   ALVs were generated
Error 617   MERGED data greater than 64K
Error 618   Multiple OVERLAY usage--previous occurrences were ignored
Error 619   Ignoring null OVERLAY list
Error 620   Missing '#' at end of OVERLAY list
Error 621   Conflicting integer sizes found
Error 622   Conflicting math types found
Error 623   Regargs function <function> called through overlay manager.
            Parameters passed in registers to this function will be
            destroyed.  Use NEWOCV option.
Error 624   Absolute reference to <symbol> module:  file <filename>
Error 625   Proper math library has not been included
Error 626   Libcode used on module <module>
Error 627   Near references found in executable that has a module
            compiled with FARONLY option
```

1.4 cplusplus

```
Error   1101   Illegal token.
Error   1102   Can't find file:  filename.
Error   1103   Invalid file name.
Error   1104   End of file encountered in comment.
Warning 1105   Invalid escape sequence.
Error   1106   illegal preprocessor directive.
Warning 1107   Extra token(s) after preprocessor directive.
Error   1108   Missing identifier in preprocessor context.
```

```
Error   1109   Redefinition of preprocessor symbol:  symbol.
Error   1110   Missing ')' in macro call.
Error   1111   Missing argument to preprocessor macro.
Error   1112   Missing comma in preprocessor expression.
Error   1113   Illegal operator in preprocessor context.
Error   1114   Missing operand in preprocessor context.
Error   1115   Illegal expression in preprocessor context.
Error   1116   Preprocessor number not a true number.
Error   1117   Integer required in preprocessor expression.
Error   1118   Extra #else or #elif.
Error   1119   Extra #endif.
Error   1121   Invalid #line format.
Error   1122   Missing parenthesis in preprocessor expression.
Error   1123   Illegal use of # operator.
Error   1124   #error directive.
Error   1125   Illegal ## expression.
Error   1126   Illegal operand of ## operator.
Error   1127   Unterminated string or character constant.
Error   1129   Character literals must contain at least one character.
Error   1130   Unterminated preprocessor conditional.
Error   1200   Syntax error more-explanation.
Error   1205   Newline within string or character literal.
Error   1206   Bad character in input (hex-number).
Warning 1208   C style comment starting on line line-number never ends.
Error   1319   'identifier' not declared.
Error   1320   No such class: 'identifier'.
Error   1321   'struct' or 'class' used on 'enum identifier'.
Error   1322   'enum' used on 'class identifier'.
Error   1323   'identifier' previously declared to be a type-name.
Error   1324   'identifier' redefined.
Error   1325   Scoped declaration in parameter list.
Error   1326   Label 'label-name' not defined.
Error   1327   Label 'label-name' previously defined.
Error   1328   Repeated keyword or type name:  'keyword'.
Error   1329   Conflicting keywords or type names:  'keyword-1' and
               'keyword-2'.
Error   1330   Must be integral, pointer, or member pointer.
Error   1331   Must be integral.
Error   1332   No such conversion.
Error   1334   Expression is not modifiable.
Error   1335   Invalid use of '&' address-of operator (object).
Error   1336   Cannot initialize (variable) with (initializer).
Error   1337   Preprocessor error.
Error   1338   Unexpected end of file.
Warning 1339   A non-lvalue array was converted to a pointer.
Error   1340   The base name 'class-1' is ambiguous in class 'class-2'.
Error   1342   Conversion from a virtual base class ('class-name') to a
               derived class is not allowed.
Error   1343   Ambiguous conversion to integral type from 'class
               class-name'.
Error   1344   Ambiguous conversion to pointer from 'class class-name'.
Error   1345   Ambiguous conversion to testable from 'class class-name'.
Error   1346   Ambiguous conversion to derived member pointer.
Error   1347   Ambiguous conversion of overloaded function pointer.
Error   1348   Ambiguous conversion to class.
Error   1349   Ambiguous conversion.
Error   1350   Ambiguous function call.
```

```
Error   1351   Overloaded functions ('function-1' and 'function-2') used
               ambiguously in conditional expression.
Error   1352   Ambiguous common base class:  class-name.
Error   1353   Ambiguous member name:  member-name.
Error   1354   Non-static member 'member-name' must be used with dot, arrow,
               or address-of operator.
Error   1355   Value of an undefined class cannot be used.
Error   1356   An array may not be the target of an assignment.
Error   1357   A function may not be the target of an assignment.
Error   1358   Cannot operation a pointer to type.
Error   1359   Typedef names cannot be declared in parameter lists.
Error   1361   Cannot take the address of a member of virtual base class.
Error   1362   Invalid initializer.
Error   1363   Invalid use of void.
Error   1364   Cast to undefined class not allowed.
Error   1365   Cannot find offset into non-class.
Error   1366   Cannot find offset into undefined class.
Error   1367   Invalid use of the scope operator.
Error   1368   Cannot find the offset of 'object'.
Error   1369   Cannot find offset because class 'class-name' has no member
               named 'member-name'.
Error   1370   Cannot take the size of an undefined class.
Error   1371   Cannot dereference pointer to undefined class.
Error   1372   No such constructor.
Error   1373   'identifier' previously declared as type-1.  Cannot be
               defined as type-2.
Error   1374   No such member 'member-name'.
Error   1375   Member 'member-name' redeclared.
Error   1376   'identifier' not a definable member.
Error   1377   'this' may occur only in a (non-static) member function.
Error   1378   Cannot create a new value of a function.
Error   1379   Cannot create a new value of a reference.
Error   1380   Cannot create a new instance of an undefined class.
Error   1382   Missing array size in expression.
Error   1383   Class 'class-name' has no default constructor.
Error   1384   Cannot initialize new array.
Error   1385   Cannot delete an object of an undefined class.
Error   1386   Length expression of array must be integral.
Error   1387   No match for call to function or overloaded operator.
Error   1388   Missing constructor body.
Error   1389   Non-virtual functions ('function-name') cannot be declared
               pure.
Error   1391   Uninitialized const identifier.
Error   1392   Uninitialized const identifier or reference:  identifier.
Error   1393   Const identifier or reference member 'member-name' must be
               initialized.
Error   1394   Member 'member-name' must have initializer, class
               'class-name' has no default constructor.
Error   1395   Base 'class-name' must have initializer, class 'class-name'
               has no default constructor.
Error   1396   Virtual base class 'class-name' must have initializer since
               class has no default constructor.
Error   1397   'identifier' is not a base class or member of class
               'class-name'.
Error   1398   Member access through protected base class not allowed for
               'member-name'.
Error   1399   Member access through private base class not allowed for
```

```
                'member-name'.
Error   1400    Base access through protected base class not allowed.
Error   1401    Base access through private base class not allowed.
Error   1402    Cannot access protected member 'member-name'.
Error   1403    Cannot access private member 'member-name'.
Error   1404    Virtual function 'function-name' declared in virtual base
                'class-name' must be overridden.
Error   1406    Parameter of type 'void'.
Error   1407    Default argument expression missing.
Error   1408    Multiple declarations of function specifying default
                arguments.
Error   1409    Arrays cannot contain elements of type 'void'.
Error   1410    Arrays cannot contain bitfields.
Error   1411    Arrays cannot contain functions.
Error   1412    Functions cannot return functions.
Error   1413    Functions cannot return arrays.
Error   1414    Functions cannot return bitfields.
Error   1415    Functions cannot return undefined classes.
Error   1416    Pointers cannot point to references.
Error   1417    Pointers cannot point to bitfields.
Error   1418    References cannot refer to references.
Error   1419    References cannot refer to bitfields.
Error   1420    References cannot refer to objects of type 'void'.
Error   1421    Member pointers cannot point to bitfields.
Error   1422    Member pointers cannot point to references.
Error   1423    Member pointers cannot point to objects of type 'void'.
Error   1424    Bitfields must be of integral type.
Error   1425    Overloaded functions with indistinguishable arguments.
Warning 1426    K&R C style function definition.
Error   1427    K&R C style functions cannot return classes with constructors
                or destructors.
Error   1428    Conversion function must be a member function.
Error   1429    Destructor function must be a member function.
Error   1430    Conversion function 'function-name' not correctly declared.
Error   1431    Destructor function 'destructor' not correctly declared.
Error   1432    Copy constructor for a class may not take an argument whose
                type is that class.
Error   1433    Operator function 'function-name' not correctly declared.
Error   1434    Invalid linkage specifier.
Error   1435    Linkage differs from prior declaration.
Error   1436    Unknown linkage convention.
Error   1437    Missing class name.
Error   1438    Repeated base class.
Error   1439    Objects of abstract classes ('object-name') cannot be
                declared.
Error   1440    Object of type 'void'.
Error   1441    Static members ('member-name') of a local class may not be
                initialized.
Error   1442    Cannot use undefined enum 'identifier'.
Error   1443    Enum constants ('identifier') must be initialized with
                integral values.
Error   1444    A class cannot be a member of itself.
Error   1445    Cannot declare members of an undefined class.
Error   1446    Cannot declare arrays of an undefined class.
Error   1447    Cannot declare variables of an undefined class.
Error   1448    Cannot initialize data members in member declaration.
Error   1449    Member function of a local class must be defined within that
```

```
                class:  class-name.
Error    1450   Member 'member-name' declared 'void'.
Error    1451   'friend' used on non-function.
Error    1452   'friend' can only be used inside a class.
Error    1453   Invalid syntax for access declaration.
Error    1454   Invalid access adjustment:  'member-name'.
Error    1455   Access cannot be changed, but only reinstated.
Error    1456   Previously declared as a member in this class.
Error    1457   'class::member' is not a member of a base class.
Error    1458   Access declaration names class that is not a base of this
                class.
Error    1460   Constructor function 'constructor' not correctly declared.
Error    1461   Destructor function 'destructor' not correctly declared.
Error    1462   Operator function 'function-name' not correctly declared.
Error    1463   Static functions ('function-name') cannot be virtual.
Error    1464   Constructors ('constructor') cannot be virtual.
Error    1465   Static functions ('function-name') cannot be used to override
                virtual functions.
Error    1467   Linkage specification cannot be used in a member declaration
                ('member-name').
Error    1468   Cannot define classes or enums in return types or parameter
                lists.
Error    1469   Invalid parameter name 'parameter'.
Error    1472   Formal 'argument' is not listed in function declaration.
Error    1473   Initialized local extern 'variable'.
Error    1474   Type names (name) cannot be initialized.
Error    1475   Class with constructors must have an initializer.
Error    1476   Cannot define classes or enums in type names.
Error    1477   Not a function.
Error    1478   A mem-initializer may be used only within constructor
                functions.
Error    1479   Base or member 'identifier' re-initialized.
Error    1480   Old style base initializer cannot be used on class with no
                bases.
Error    1481   Old style base initializer cannot be used on class with
                multiple base classes.
Warning  1482   Statement is unreachable.
Error    1483   'case' label must be within a switch statement.
Error    1484   'default' label must be within a switch statement.
Error    1485   'continue' must be within a loop ('do', 'for', or 'while')
                statement.
Error    1486   'break' must be within a switch or loop ('do', 'for', or
                'while') statement.
Error    1487   Missing return value.
Error    1489   Return value given for constructor, destructor, or void
                function.
Error    1490   Missing function name in function declaration.
Error    1491   Illegal formal declaration list in prototype function
                definition.
Error    1492   Formal ('argument') must be declared in function header
                identifier list.
Error    1493   Expression in array declarator must be constant expression.
Error    1494   Expression in array declarator must be integral.
Error    1495   Expression in array declarator must be positive.
Error    1498   Invalid bitfield size.
Error    1499   Cannot use undefined class 'class-name' as base class.
Error    1500   Missing declaration-specifier.
```

```
Error   1501   Illegal use of 'item' in local member function.
Error   1502   A class cannot be derived from a union ('union-name').
Error   1503   A union ('union-name') cannot be derived from another class.
Error   1504   Constant expression contains a division by zero (0).
Error   1506   Cannot take the size of a function.
Error   1507   Cannot take the size of a bitfield.
Error   1508   Cannot take the size of void.
Error   1509   Cannot take the size of array with unspecified length.
Warning 1510   Cannot jump into a block to a label after a declaration
               having an initializer.
Error   1511   Overloaded member functions ('function-name') may not be both
               static and non-static.
Error   1512   Function hides a virtual function from base class.
Error   1513   Overriding virtual function has different return type.
Error   1514   Arrays cannot contain references.
Error   1515   Previous declaration of function had different return type.
Error   1516   Cannot have two extern "C" functions with same name ('name').
Error   1517   Previous declaration differed in the use of __builtin.
Error   1518   object-type ('expression') cannot be used in default argument
               expressions.
Error   1522   Keyword can only be used on functions.
Warning 1523   'keyword' cannot be applied to object-type.
Error   1524   Previous declaration was not static.
Error   1525   Function declared 'inline' after first use.
Error   1528   Member functions must be C++ functions.
Error   1530   Previous errors prevent continuation.
Error   1531   A declaration must declare something.
Error   1532   function-name cannot have 'storage-type' storage class.
Warning 1533   Extra comma at end of enumeration list.
Warning 1534   Enum value:  value is used for both 'enum-1' and 'enum-2'.
Error   1535   Cannot overload 'main'.
Error   1536   Cannot call or take the address of 'main'.
Error   1537   'main' cannot be 'storage-type'.
Error   1538   Anonymous classes cannot have constructors or destructors.
Error   1539   Destructor names ('destructor') must be the same as their
               class name ('class').
Error   1540   Expression in array declarator must not be negative.
Error   1541   Cannot allocate array of class 'class-name' with no default
               constructor.
Error   1542   Invalid constructor given for member 'member-name'.
Error   1543   'operand-1' and 'operand-2' are not compatible types for
               conditional operator.
Error   1544   (type-1) operator (type-2):  Invalid type for binary
               operator.
Error   1545   'operand' is of invalid type for postfix operator 'operator'.
Warning 1546   'operator' is invalid for operand type 'operand'.
Error   1547   'object' is of invalid type for call operator.
Error   1548   Invalid pointer conversion from 'type-1' to 'type-2'.
Warning 1549   Non-const and/or non-volatile member function called with
               const and/or volatile object.
Warning 1550   Non-constant reference 'reference-object' initialized with a
               non-lvalue.
Error   1551   Cannot take size of pointer to overloaded function
               'function-name'.
Error   1553   Error writing to output file:  filename.
Error   1554   Inline member function does not end.
Error   1555   Static function 'function-name' was not defined.
```

```
Error   1556   Global anonymous unions must be static.
Error   1557   Anonymous unions may not have function members.
Error   1558   Anonymous unions may not have private or protected members.
Error   1559   'identifier' redeclared in anonymous union.
Error   1560   An anonymous union cannot be declared as a static member.
Error   1562   Conflicting declaration of name 'identifier' reserved for
               purpose.
Error   1564   Cannot initialize a function ('function-name').
Error   1565   Static members (member-name) cannot be initialized by a
               mem-initializer.
Error   1566   Enum constants (identifier) cannot be initialized by a
               mem-initializer.
Error   1567   Types must match in a delete expression:  type-1->~type-2.
Error   1568   Cannot create a new value of a void.
Error   1569   Loop in -> operators.
Error   1570   A linkage-specification may occur only in file scope.
Error   1571   Cannot define a type in return or argument types.
Error   1572   object may not have the same name as its class.
Error   1573   An overloaded operator cannot have default arguments.
Error   1574   Invalid use of abstract class:  class-name.
Error   1575   An object of a class with a object-type may not be a member
               of a union.
Error   1576   Error declaring 'new':  reason.
Error   1577   Error declaring 'delete':  reason.
Error   1578   Initializer-clause cannot be used for class having a
               object-type.
Error   1579   Conversion to a virtual base class ('class-name') from a
               derived class is not allowed for member pointers.
Error   1580   Cannot return (attempted-return-type) from function returning
               (declared-return-type).
Error   1581   Function 'function-name' has an initializer.
Error   1582   Character array (array-name) too short for string of length
               (string-length).
Error   1583   Too many initializers for (array-name):  found n
               initializers.
Error   1584   Too many initializers for (class-name).
Error   1585   Left operand of 'operator' must be type.
Error   1586   Type 'type' is invalid for the left operand of 'operator'.
Error   1587   Case label value must be a constant expression.
Error   1588   Duplicate case label value.
Error   1589   More than one default.
Error   1590   symbol-name is not an enum.
Error   1591   symbol-name is not a class, struct, or union.
Warning 1592   Wide and narrow character strings concatenated, using width.
Warning 1593   Missing return statement.
Warning 1594   Zero-length array used.
Warning 1597   '%s' assigned to '%s'.
Error   1610   Previous declaration of 'symbol' was 'attribute', this
               declaration is 'attribute'.
Error   1611   Previous declaration of 'symbol' differed in the use of
               'keyword'.
Error   1612   Asm function has parameter without a register.
Error   1613   Asm function uses register '%s' more than once.
Error   1614   Previous declaration of asm function used different
               registers, was '%s', now '%s'.
Error   1615   Explicit register 'register' keyword used in non-asm
               function.
```

```
 Error   1616   Vararg functions cannot be '__asm'.
```

1.5 strict

```
 Using the STRICT compiler flag  automatically turns on the ANSI flag
as well as some additional warnings.  When the STRICT flag is used, the
following warning messages are turned on:

 Warning  18   non-ANSI use of operator in preprocessor condition
 Warning  51   C++ comment detected
 Warning  70   unrecognized escape sequence
 Warning 108   zero-length arrays are not an ANSI feature
 Warning 111   non-portable enum type specified
 Warning 120   Integral type mismatch:  possible portability problem
                Expecting "<type1>", found "<type2>"
 Warning 137   ANSI limits #line numbers to between 1 and 32767
 Warning 149   incomplete struct/union/enum tag in prototype scope "<name>"
 Warning 159   use of unary minus on unsigned value
 Warning 162   non-ANSI use of ellipsis punctuator
 Warning 163   initialization of auto struct, union, or array
 Warning 164   & applied to array
 Warning 176   implicitly promoted formal "<name>" conflicts with prototype
                See line <number> file "<filename>"
 Warning 178   indirect call without indirection operator
 Warning 179   narrow type used in old-style definition
 Warning 180   no space between macro name and its replacement list
 Warning 187   negative value assigned to unsigned type
 Warning 189   <option> option differs from the one used to build the GST
 Warning 213   empty argument to preprocessor macro
 Warning 220   old-fashioned assignment operator taken as "<operators>"
```

1.6 ANSI COMPILER FLAG

```
 Using the ANSI compiler flag  turns on a subset of the messages turned
on by the STRICT flag.  The ANSI flag turns on the following warning
messages:

 Warning  18   non-ANSI use of operator in preprocessor condition
 Warning  51   C++ comment detected
 Warning  70   unrecognized escape sequence
 Warning 108   zero-length arrays are not an ANSI feature
 Warning 111   non-portable enum type specified
 Warning 137   ANSI limits #line numbers to between 1 and 32767
 Warning 162   non-ANSI use of ellipsis punctuator
 Warning 176   implicitly promoted formal "<name>" conflicts with prototype
                See line <number> file "<filename>"
 Warning 180   no space between macro name and its replacement list
 Warning 213   empty argument to preprocessor macro
```

1.7 scmxx1

```
*** Can't delete old GST:  object is in use
    Continuing with no GST file
```

You specified the MAKEGST option, but an existing copy of the same GST was in use by another program. Check for other compilations or applications that are using the GST. You may also be browsing the GST with the hypergst utility.

1.8 scmxx2

```
*** Can't open GST file:  <gst-filename>
```

The specified GST file could not be loaded. Either the file is an invalid GST file, or the file does not exist.

1.9 scmxx3

```
*** Can't open sc:libs/<lib-name>.library
```

The specified shared library could not be found. Make sure the library is available in sc:libs.

1.10 scmxx4

```
*** Can't open <type> file "<name>" for <mode>
```

The compiler could not open the specified file. The mode is either read or write.

1.11 scmxx5

```
*** Combined output filename too long
```

The filename produced by the compiler, with the path, overflowed the compiler's internal buffer (255 bytes).

1.12 scmxx6

```
*** CXERR: num
```

An internal error prevented the compiler from continuing. Please contact the Technical Support Division.

1.13 scmxx7

```
*** CXWRN: text
```

An internal error occurred. With CXWRN errors, the compiler attempts to continue the compilation, but may not be able to do so. Please contact the Technical Support Division.

1.14 scmxx8

```
*** Freeing Resources
```

If you compile your program and your machine runs low on memory, the compiler displays this message and frees memory to enable it to continue the compilation. You can force the compiler to free memory at any time by pressing Control-F in the window to which the compiler is sending output.

1.15 scmxx9

```
*** Floating point overflow optimizing constants
```

The global optimizer was attempting to perform compile-time constant calculations, but the calculations caused a floating-point error. Your code is causing floating-point numbers to overflow.

1.16 scmx10

```
*** Invalid symbol definition: symbol-name
```

You attempted to define the specified symbol on the command line with the DEFINE compiler option, but the symbol did not adhere to the normal rules for C preprocessor symbol syntax.

1.17 scmx11

```
*** I/O error <code> on file "name"
```

The compiler received the specified I/O error code from the operating system while attempting to read the named file. Refer to The AmigaDOS Manual, 3rd Edition (Commodore-Amiga, Inc. 1991) or see the header file dos/dos.h for details on the numeric error codes.

1.18 scmx12

```
*** Seek error on object file
```

The compiler attempted to perform a seek operation on the output object file but encountered an I/O error.

1.19 scmx13

```
*** Warning:  Debugging information may be incorrect for optimized code.
```

You are compiling with the debug option, and you are also using the global optimizer. The optimizer eliminates variables and moves functions inline, so the debugger may not be able to provide accurate information.

1.20 scm001

```
Warning 1:  invalid preprocessor command
```

This warning is generated by invalid use of preprocessor commands. For example, you could specify an unrecognized command, fail to include a space between command elements, or use an illegal preprocessor symbol. The command is ignored and compilation continues.

1.21 scm002

```
Error 2:  unexpected end of file
```

This error is generated when the compiler expects more data, but it encounters the end of an input file. This error may occur in a #include file or in the original source file. A missing #endif or unbalanced curly brace or parentheses in the source file or in one of the previously included files may produce this message. In many cases, correcting a previous error eliminates this error.

1.22 scm003

```
Error 3:  file not found "filename"
```

The filename specified in a #include command was not found or could not be opened.

1.23 scm004

```
Error 4:  invalid lexical token
```

A character was found in the file that is not a standard character in the C character set or is in an inappropriate place. For example, entering a pound sign (#) in the middle of non-preprocessor C code or entering nonprintable control characters anywhere except in a comment produces this error.

1.24 scm005

```
Error 5:  invalid usage for macro "macro-name"
```

Your code invoked a macro incorrectly. Check for unbalanced parentheses and other syntax errors. The problem may be in a macro used by the macro that you invoked in your program.

1.25 scm006

```
Error 6:  line buffer overflow
```

A line of preprocessed input was longer than the line buffer size. The size of the line buffer is controlled by the ppbuf and memsize options. If you do not specify ppbuf or memsize, the size of the line buffer is 8192 bytes.

1.26 scm007

```
Warning 7:  register parameters require a prototype
            Stack parameters used
```

You compiled with parameters=register or parameters=both, but did not provide a prototype for a function that you called. Without a prototype, the compiler cannot pass parameters in registers so it passes parameters on the stack instead.

Omitting a prototype can cause problems when your program is linked. The compiler identifies functions that expect arguments in registers by placing an at sign (@) in front of the function name. The @ replaces the underscore that the compiler normally places at the beginning of function names. Because you compiled with parameters=register, the compiler prepends the @ sign to the function name when the function is defined. However, because you did not provide a prototype, the compiler assumes the function is called with stack parameters and when the function is called, the caller prepends an underscore (_) instead of the @ sign. The @ version of the function will not satisfy the reference to the _ version, so the linker will issue an error saying that the @ version of the function is undefined.

1.27 scm008

```
Error 8:  invalid conversion
```

You attempted to cast a type to an incompatible type. This error usually occurs when you attempt to convert something into an array, a structure, or a function. Check for missing indirection (*) and/or address (&) operators. For example, the following program tries to assign a whole structure to a pointer.

```
void main(void)
{
   struct FOO
   {
      int a, b;
   } f;
   struct FOO *p;
   p = f;  /* Error 8 */
   p = &f; /* Correct */
}
```

1.28 scm009

```
Error 9:  undefined identifier "name"
```

The specified identifier has not been declared. You may not have included the proper header files to declare an extern, or you may have misspelled the name of a variable. This message is produced only once for each undeclared identifier. Subsequent uses of the identifier do not produce a message. Subsequent declarations of the identifier may produce messages about redeclaring the variable. Fix the error that is causing the first error 9 message and recompile your program before trying to fix additional messages involving the same variable.

1.29 scm010

```
Error 10:  invalid subscript expression
```

An error was detected in an expression used inside square brackets ([]). This error may occur if:

- the expression is missing
- the expression is a preprocessor macro that evaluates to nothing
- the result of the expression is void
- the result of the expression is a pointer, a structure, or a union.

1.30 scm011

```
Error 11:  string not terminated

The closing double quote (") was not provided when defining a string.
```

1.31 scm012

```
Error 12:  invalid structure reference

The operand preceding the structure member (.) or structure pointer (->)
operator is not a structure or a pointer to a structure, respectively.
Make sure you are not trying to reference a structure member with the
structure pointer operator or a structure pointer with the structure member
operator.  In many cases, correcting a previous error eliminates this
message.
```

1.32 scm013

```
Error 13:  member name missing

The name of the desired structure or union member did not follow the
structure member operator (.) or the structure pointer operator (->).
Check for preprocessor macros that may be defined to the same name as the
member.
```

1.33 scm014

```
Error 14:  undefined member "name"

The indicated identifier is not a member of the structure or union to which
the structure member operator (.) or the structure pointer operator (->)
referred.
```

1.34 scm015

```
Error 15:  invalid function call

An identifier or constant is used where a function or function pointer
identifier is required.  This message can occur if you attempt to use a
variable not declared as a function or function pointer but give the
variable a parameter list.  The compiler sees the parenthesized expression
and thinks you are calling a function.  Check for typographical errors such
as leaving out an operator in an expression, which produces a variable and
a parenthesized expression.  Such typographical errors may also occur in
preprocessor macros.  For example:

  /* The incorrect expression below generates an */
  /* "invalid function call" error.              */
```

```
int i, j;
void function(void)
{
   i = i + (j*2); /* Intended expression */
   i = i   (j*2); /* INCORRECT - deleted "+" operator */
}
```

1.35 scm016

Error 16: invalid function argument

A function argument expression following the left parenthesis of a function call is invalid. You may see this message if you omit:

- an argument expression
- a right parenthesis from a function call
- a comma separator between two function arguments.

For example:

```
func(.);
func(;
func(1 2);
```

1.36 scm017

Error 17: too many operands

During expression evaluation, the end of an expression was encountered, but more than one operand was still awaiting evaluation. This message may occur if an expression contained an incorrectly specified operation.

1.37 scm018

Warning 18: non-ANSI use of operator in preprocessor condition

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=18 options. The ANSI C Standard states that the sizeof and comma (,) operators should not be used in preprocessor conditions. The SAS/C Compiler supports their use in preprocessor conditions, but programs that require strict adherence to the ANSI C Standard should not use them.

1.38 scm019

```
Error 19:  unbalanced parentheses
```

The number of opening parentheses in an expression did not equal the number of closing parentheses. If the expression appears correct in the C source file, check any preprocessor macros to make sure they generate balanced parentheses.

1.39 scm020

```
Error 20:  invalid constant expression
```

An expression that did not evaluate to a constant was encountered in a context that required a constant result. The compiler must be able to evaluate any constant expression (for example, expressions used to initialize static or external data) when the program is compiled. The expression in question did not meet this criterion. You may have used an illegal operator for a constant expression (such as ++, +=, function calls, and so on), or you may have used a variable whose value is available only at run time.

1.40 scm021

```
Error 21:  illegal use of struct, union, or array type
```

An identifier declared as a structure, union, or array was encountered in an expression where such types are not permitted. For example, you cannot use the ++ postincrement operator with a structure:

```
struct FOO
{
   int a, b, c;
} f;

 f++;      /* Error 21 */
```

1.41 scm022

```
Error 22:  __asm functions cannot accept structure or union arguments
           Use pointers instead
```

An attempt was made to pass an instance of a structure or union to a function declared with the __asm keyword. You must only pass pointers to __asm functions.

1.42 scm023

```
Error 23:  invalid use of conditional operator (?:)
```

The conditional expression operator (?:) was used incorrectly. You may have included the question mark (?) but left out the colon (:). Also, check for an invalid expression after the operator.

1.43 scm024

```
Error 24:  pointer operand required
```

An expression required a pointer at a specific place, but a non-pointer operand was provided. The compiler may generate this message if an expression after the indirection operator (*) was not a pointer or array expression or if the expression before the array indexing operators ([]) was not a pointer or array expression.

1.44 scm025

```
Error 25:  modifiable lvalue required
```

You have attempted to assign a value to an expression that cannot be modified. An lvalue is any expression that can appear on the left side of an assignment operation. For example, the ANSI C Standard states that the result of a cast is not an lvalue; therefore, the following statement is invalid:

```
long x;
(short)x = 2;  /* Error 25 */
```

The following examples also generate this error message:

```
#define ADDONE(x) (x)++
ADDONE(12);     /* Error 25: Cannot increment a constant */
ADDONE(&g);     /* Error 25: Cannot assign to an address */

if(func(10)=j-2);  /* Error 25: "==" was intended, not "=" */

&x = &y;        /* Error 25: Cannot assign to an address */
```

You may have defined a variable with the const keyword and then tried to modify the value of that variable. The SAS/C Development System Library Reference contains many variables defined with the const keyword, especially pointers to strings. This keyword is required by the ANSI C Standard and indicates that these variables will not be modified in the library function to which they are passed. The const keyword is present in the parameter list of the prototype that is found in the associated header file. This prototype, not the declaration you should include in your program, is what is described in the synopsis for each function in the SAS/C Development System Library Reference. Use the synopsis as a guideline for your declarations, but do not include the const keyword.

1.45 scm026

```
Error 26:  arithmetic operand required
```

An expression required an arithmetic operand, but the provided operand was not arithmetic. An operand is arithmetic if it declared as char, short, int, long, float, or double or the signed and unsigned variants of these types.

Pointers, structures, unions, and functions are not arithmetic operands.

1.46 scm027

```
Error 27:  arithmetic or pointer operand required
```

An expression required an arithmetic or pointer operand, but a structure or union was provided. An operand is arithmetic if it declared as char, short, int, long, float, or double or the signed and unsigned variants of these types.

A pointer operand can be a pointer to any other data type, or it can be the address of a variable or function.

1.47 scm028

```
Error 28:  missing operand
```

During expression evaluation, the end of an expression was encountered but not enough operands were available for evaluation. The compiler may generate this message if you specified a binary operator (such as the addition, subtraction, multiplication, or division operator) with only one operand. Also, check for invalid preprocessor macro expansions. For example:

```
int i;
int ary[10];

i = i + ;     /* Error 28 */
i = ary[i*]4; /* Error 28 (Among others) */
```

1.48 scm029

```
Error 29:  operation cannot be performed on a pointer
```

An operation was specified that was invalid for pointer operands, such as one of the arithmetic operations other than addition or subtraction.

1.49 scm030

```
Warning 30:  pointers do not point to same type of object
```

In an assignment statement defining a value for a pointer variable, the expression on the right side of the assignment (=) operator did not evaluate to NULL or to a pointer of the same type as the pointer variable on the left side of the assignment operator. The warning is also produced when a pointer of any type is assigned to an arithmetic object.

1.50 scm031

```
Error 31:  integral operand required
```

An expression required a given operand to be an integral type, but the actual operand was not an integral type. An operand is integral if it is declared as char, short, int, or long or the signed and unsigned variants of these types.

For example, the following code generates error 31:

```
double d;
int ary[10];

ary[d] = 10;  /* Error 31 */
```

1.51 scm032

```
Error 32:  cannot convert to required type
```

The compiler was unable to convert a data item from its base type to the type required by the operation. The compiler may generate this message if you attempt to cast any data type to a structure, instead of casting the type to a pointer to a structure, or if you attempt to cast a structure to any other type. This message can also be produced for implied conversions, such as passing a structure as a parameter to a function expecting some other type. For example, the following code generates error 32:

```
struct FOO f;
int j;

j = (int)f;  /* Error 32 */
```

1.52 scm033

```
Warning 33:  non-portable operation on structure or union
```

Your code has attempted to use an operator on a structure or union that is illegal for that type. For example, you may have used the equality ==

operator on two structures. The ANSI C Standard does not permit the use of relational operators on structures or unions.

The SAS/C Compiler generates the equivalent of a memcmp call for this construct, but it may not perform as expected. Because structures may contain padding bytes, two structures of the same type with all identical members may compare false. If you intend to use direct structure comparison, make sure you declare the structure static or extern, or initialize the structure to zeroes using a call to memset.

For example, the following code generates warning 33:

```
#include <proto/dos.h>
struct FileInfoBlock fib1, fib2;

if(fib1 == fib2)  /* Warning 33 */
```

1.53 scm034

Error 34: invalid initializer expression

The expression used to initialize an object was invalid. The compiler may generate this message if you fail to separate elements in an initializer list with commas or if you attempt to initialize an array to a single object, as shown in the following examples:

```
int a[3] = 0;          /* Error 34 */
int b[3] = { 1 2 3 };  /* Error 34 */
```

1.54 scm035

Error 35: closing brace expected

The compiler expected a closing brace (}) to terminate the definition of a function, structure, or nested block scope, but the brace is missing. The compiler may generate this message if:

- too many elements occur in an initializer expression list
- a structure member was improperly declared
- the end of the source file is reached before a definition is complete
- a previous error occurred in a control statement.

1.55 scm036

Warning 36: control cannot reach this statement

A statement with no label followed a goto, return, break, or continue statement. The statement is therefore unreachable. This warning can sometimes be produced incorrectly if the compiler reported a previous error

while in a control flow statement. Fix all previous errors and recompile your program before trying to fix the error that is generating this message.

1.56 scm037

```
Error 37:  duplicate statement label "name"
           See line number file "filename"
```

The specified statement label has already been defined in the current function. You cannot define the same label more than once in the same function.

1.57 scm038

```
Error 38:  unbalanced braces
```

In a set of compound statements, the number of opening left braces ({) did not equal the number of closing right braces (}). This error may be produced incorrectly if the compiler reported a previous error in a control flow statement. Fix any previous errors and recompile your program before trying to fix the error that is generating this message.

1.58 scm039

```
Error 39:  invalid use of keyword "keyword"
```

One of the C language reserved words appeared in an invalid context (for example, as a variable name).

1.59 scm040

```
Error 40:  break not inside loop or switch
```

A break statement was detected that was not within the scope of a while, do, for, or switch statement. This error may be produced incorrectly because of errors in previous statements.

1.60 scm041

```
Error 41:  case not inside switch
```

A case prefix was encountered outside the scope of a switch statement. This error may be produced incorrectly because of errors in previous statements.

1.61 scm042

```
Warning 42:  case expression not integral
```

The expression defining a case value did not evaluate to an integral constant. This message is generated as an error message if the expression could not be converted into an integral constant and as a warning if the expression could be converted into an integral constant. For example, if you use a variable as a case value, the compiler generates an error message. If you use a floating-point constant as a case value, the compiler converts the constant to an integer (thereby truncating its value) and generates a warning message.

For example, in the following code, the case value 1.6 is truncated to 1, and the value -1.6 is truncated to -1.

```
switch(i)
{
   case 1.6:  /* Warning 42 */
      break;

   case -1.6:  /* Warning 42 */
      break;
}
```

1.62 scm043

```
Error 43:  duplicate of case value
           See line number file "filename"
```

You have used the same case value more than once within the same switch statement. Check for possible preprocessor macro definitions that expand to the same value.

For example, both of the following case statements evaluate to zero:

```
#define FOO 0
#define BAR 2

switch(i)
{
   case FOO:
      break;
   case (FOO*BAR):  /* Error 43 */
      break;
}
```

1.63 scm044

```
Error 44:  continue not inside loop
```

A continue statement was detected that was not within the scope of a while,

do, or for loop. This error may be produced incorrectly because of errors in previous statements.

1.64 scm045

```
Error 45:  default not inside switch
```

A default label was encountered outside the scope of a switch statement. This message may be produced incorrectly because of errors in previous statements.

1.65 scm046

```
Error 46:  duplicate default
           See line number file "filename"
```

A default label was encountered within the scope of a switch statement in which a default label had already been encountered.

1.66 scm047

```
Error 47:  while missing from do statement
```

A while clause did not follow the body of a do statement. This message may be produced incorrectly because of errors in previous statements.

1.67 scm048

```
Error 48:  invalid while expression
```

The expression defining the looping condition in a while or do loop was void or was missing. If you intend for a loop to be infinite, you must supply a constant (such as 1) for the while condition. The error may be produced because of a preprocessor macro that expands to invalid code. In many cases, fixing a previous error eliminates this message.

1.68 scm049

```
Error 49:  else not associated with if
```

An else keyword was detected that was not in the scope of a preceding if statement. This message may be caused by an error in a preceding statement, especially if the previous error occurred while processing the if statement with which the else statement is associated.

1.69 scm050

```
Error 50:  label missing from goto
```

The compiler expected a statement label to follow the goto keyword but the label was missing. This message may be produced incorrectly because of errors in previous statements.

1.70 scm051

```
Warning 51:  C++ comment detected
```

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=51 options. In C++, comments begin with two slashes (//) and terminate at the end of the line on which they begin. The SAS/C Compiler accepts comments entered in this way for convenience and for compatibility with other implementations, but they are not part of the ANSI C Standard.

1.71 scm052

```
Error 52:  invalid if expression
```

The expression following the if keyword on an if statement was void, invalid, or missing. This error may be caused by a preprocessor macro expanding to inappropriate values or may be the result of an expression with an invalid result type (such as a structure type). In many cases, fixing a previous error eliminates this message.

1.72 scm053

```
Error 53:  invalid return expression
```

The expression following the return keyword was void, invalid, or missing. The compiler may generate this message if a preprocessor macro expands to inappropriate values. In many cases, fixing a previous error eliminates this message.

1.73 scm054

```
Warning 54:  switch expression not integral
```

The expression defining the test value for a switch statement did not define an integral value as required by the ANSI C Standard. The value supplied is converted to int before any attempt is made to use it. If the switch value is a floating-point value, this conversion may truncate the value. This warning can also be generated as an error if the value could not be converted to int.

1.74 scm055

```
Warning 55:  no case values for switch statement
```

The statement defining the body of a switch statement did not define any case statements. This warning may be produced incorrectly because of previous errors.

1.75 scm056

```
Error 56:  colon expected
```

The compiler expected but did not find a colon (:). This message may be generated if a case expression was improperly specified or if the colon was omitted following a label to a statement. Because the compiler scans past newlines, blanks, and comments looking for the colon, this message is usually produced at the beginning of the line following the actual error.

1.76 scm057

```
Error 57:  semi-colon expected
```

The compiler expected but did not find a semi-colon (;). This error can be generated if you use too many right parentheses or right curly braces (}). Because the compiler scans past spaces and tabs looking for the semicolon, this message is usually produced at the beginning of the line following the actual error.

1.77 scm058

```
Error 58:  missing parenthesis
```

A required parenthesis is missing. This error is often caused by previous errors.

1.78 scm059

```
Error 59:  invalid storage class
```

Possible storage classes are denoted by the keywords __near, __far, __chip, register, auto, extern, and static. Some of these keywords are invalid for certain types of data. For example, you cannot declare an external variable with the register keyword, and you cannot declare an automatic (local) variable with the __chip, __near, or __far keywords. Your code attempted to use a storage class keyword incorrectly. This error often occurs because of previous errors.

1.79 scm060

```
Error 60:  incompatible struct, union or array types
```

Incompatible structure, union, or array types were used in an expression.

```
struct A {int x;}    a;
struct B {double d;} b;
a = b;  /* Error 60 */
```

1.80 scm061

```
Error 61:  undefined struct/union tag "tag-name"
```

Your code has used a structure or union tag that has not been declared. Check for misspelled structure names. You may want to compile your program with the pponly option or the list option and look at the output produced.

This message is produced as an error if you attempt to refer to the members of an undefined structure or union and as a warning if you only use pointers to the structure or union. The warning is suppressed by default, but you can enable it with the warn=61 option. The error cannot be suppressed.

1.81 scm062

```
Warning 62:  constant number out of range for type "type"
             Valid range is low to high
```

The constant value indicated is not in the range of possible values for the type to which the value is being assigned. This message may occur if you are assigning a hexadecimal constant with the uppermost bit set to a signed variable. By definition, a hexadecimal constant is a positive value; therefore, the assignment to the variable reinterprets the constant as a negative number.

For example, the following lines generate warning 62:

```
signed char c = 0x80;    /* Warning 62 */
short s = 100000;        /* Warning 62 */
unsigned short uc = -1;  /* Warning 62 */
```

Unless you use the unschar compiler option, variables of type char are signed and produce this warning if you assign any constant to them with the high bit set (that is, any value between 128 and 255.) You can suppress this warning for a specific case by casting the constant to the appropriate type.

Out-of-range constants can cause problems in your code that are hard to debug. For example, the following code does not behave as intended:

```
int i = 0xff;
signed char c = 0xff;  /* Warning 62 */

if(i == c)
{
   /* Not executed */
}
```

The constant initializer 0xff is the decimal number 255. When assigned to the integer variable i, this value is preserved, and i gets the value 255. When assigned to the signed character variable c, the value of c becomes -1 because a signed character cannot represent numbers higher than 127, and the constant 0xff overflows. Therefore, the comparison in the if statement results in the value false.

1.82 scm063

```
Warning 63:  item "name" already declared
             See line number file "filename"
```

The named item was previously declared at the cited location. This warning is produced when two different members of the same structure, union, or enum are given the same name.

1.83 scm064

```
Error 64:  structure contains no members
```

A structure declaration did not contain any members. This error can be produced by errors encountered during the structure's declaration. Fixing a previous error may eliminate this message.

1.84 scm065

```
Error 65:  invalid function definition
```

Your code tried to define a function body inside another function body, inside a structure declaration, or inside a list of static initializers. This message may be produced incorrectly because of errors in previous statements.

1.85 scm066

```
Error 66:  invalid array limit expression
```

The expression defining the size of the subscript in an array declaration did not evaluate to a positive integral constant.

For example:

```
int x[10.7];
```

1.86 scm067

Error 67: illegal object

Your code attempted to define an illegal item. For example, you may be declaring an array of functions (instead of an array of function pointers) or a function that returns an array (instead of a function that returns a pointer). You may also be attempting to declare something other than a function as type void.

For example:

```
void x;  /* Error 67 */
```

1.87 scm068

Error 68: illegal object for structure

A structure (or union) included a function as a member. You cannot include a function as a member of a structure or union, although you can include a function pointer.

1.88 scm069

Error 69: struct name includes instance of self

The named structure or union contains an instance of itself. Although it is legal for a structure or union to include a pointer to its own type, the structure or union cannot contain an instance of itself. If the structure or union does not have a name, the name field is not printed in the message.

For example, the following code generates error 69:

```
struct FOO
{
   int a, b, c;
   struct FOO x;  /* Error 69 */
};
```

1.89 scm070

```
Warning 70:  unrecognized escape sequence

By default, this message is suppressed, but you can enable it with the
strict, ansi, or warn=70 options.  Escape sequences in string and character
constants begin with a backslash (\) and contain one or more characters
after the backslash.  The ANSI C Standard defines some escape sequences and
reserves others for future expansion and for implementation-defined
extensions.  The SAS/C Compiler ignores the backslash on any such undefined
escape sequences.  Other compilers may take different action.

For example, the following line prints the character q followed by a
newline (\n) to stdout:

   printf("\q\n");  /* Warning 70 */

Other ANSI-conforming compilers may substitute any other character for the
\q escape sequence.
```

1.90 scm071

```
Error 71:  formal declaration error "name"

A variable was declared before the left curly brace of an old-style
function definition, but the variable did not appear in the list of
identifiers in parentheses following the function name.

For example, the declaration of j in the following code generates error 71:

   int func(i)
   int i;
   int j;  /* Error 71 */
   {
   }

You may have misspelled one of your formal parameters or forgotten to add
the parameter to the parameter list.
```

1.91 scm072

```
Error 72:  conflict with previous declaration
           See line number file "filename"

A variable or function was declared that conflicts with a previous
declaration for the variable or function in the same scope.  The message
indicates the filename and line number of the original declaration.  If no
prototype exists for a function, the first use of that function implicitly
declares the function as returning an int.  If the actual definition of the
function follows the first use of the function and declares a different
return type, the compiler produces this message.

This message is produced as a warning message instead of an error message
```

if the only difference between the declarations is that one is a function with the const keyword on a parameter and the other does not have the const keyword on that parameter. Error 72 may be produced incorrectly if you forget to declare a variable in an earlier location and, therefore, received a previous error message about an undefined identifier with the same name. Fixing the previous error may eliminate this message.

1.92 scm073

Warning 73: declaration expected

The compiler expected to find the declaration of a data object or function but did not. This message can also occur if you enter too many or too few curly braces. This message may be produced incorrectly because of errors in previous statements. Fixing the previous errors may eliminate this message.

1.93 scm074

Warning 74: initializer data truncated

A static initializer expression contained more elements than the data item being initialized, as in the following example:

```
char b[3] = "abcd";    /* Warning 74 */
```

String constants always have an implied NULL byte at the end. This byte is not counted when producing this warning. If you have a character array with three elements, you may initialize it as follows:

```
char b[3] = "abc";
```

The three elements receive the values a, b, and c. If there is room, the NULL terminator byte is also copied:

```
char c[4] = "abc";
```

The elements of the above array are assigned the values a, b, c, and \0.

1.94 scm075

Error 75: invalid sizeof expression

The expression passed to the sizeof operator was invalid. This message may be produced if an attempt is made to take the size of a function, bitfield, or incomplete type, or if the result of the expression used is of type void.

For example, in the following code, a is an incomplete type because its

size is not specified when it is declared:

```
extern int a[];
int i;

i = sizeof(a);  /* Error 75 */
```

1.95 scm076

Error 76: left brace expected

The compiler expected but did not find an opening left brace. For example, you may have omitted the opening brace on a list of initializer expressions for an aggregate.

1.96 scm077

Error 77: identifier expected

The compiler expected to find the name of an identifier to be declared. The compiler may generate this message if the prefixes to an identifier in a declaration (parentheses and asterisks) are incorrectly specified or if a sequence of declarations is listed incorrectly, as in the following example:

```
int ;  /* Error 77 */
```

1.97 scm078

Error 78: undefined statement label "name"

A goto statement referred to the named label, but the label does not exist in the function that referred to it. Check the spelling of your label and the goto reference. Make sure the label is in the same function as the goto statement.

1.98 scm079

Warning 79: duplicate of enumeration value on line line

When declaring an enumeration type, more than one enumeration constant was assigned the same value, as in the following example:

```
enum COLORS
{RED=1, BLUE=2, GREEN=1};  /* RED and GREEN are identical */
```

Any enumeration constants that are not assigned an explicit value are

assigned values one higher than the previous constant in that enumeration. If the constant is the first constant in that enumeration, it is assigned the value zero. Therefore, the following enum generates warning 79:

```
  enum COLORS {RED, BLUE, GREEN=1};  /* Warning 79 */
```

RED is the first constant, so it is assigned a value of 0. BLUE is assigned a value of 1. GREEN is explicitly assigned a value of 1, which conflicts with BLUE.

1.99 scm080

```
Warning 80:  invalid bit field or misplaced ':'
```

This warning is commonly produced if you type a colon (:) when a semicolon (;) was expected. This warning can also be produced if you are actually declaring a bitfield and give an improper expression for the number of bits in the bitfield.

1.100 scm081

```
Error 81:  preprocessor symbol loop
           macro expansion too long
```

When using the oldpp compiler option, a preprocessor symbol expanded to a value that contains a circular reference back to the symbol itself, thereby creating an infinite loop in the preprocessor. Check the definition of the macro being expanded on the line in question. This message cannot occur if the oldpp compiler option is not used because the ANSI C Standard prohibits the expansion of preprocessor macros that occur as a result of expanding a previous instance of the same macro.

1.101 scm082

```
Error 82:  maximum object/storage size exceeded
           Size limit for this class is n
```

Your code attempted to declare a data item that exceeded the maximum size of objects in its storage class, or the last object declared caused the total size of declared objects in that storage class to exceed the maximum, as in the following example:

```
   char __near a[30000];
   char __near b[30000];
   char __near c[30000];
```

Because there is a limit of 65536 bytes on the amount of near data allowed, the above request for 90000 bytes of near data will not work. If you are compiling with data=near (the default), the __near keyword is implied on all data items. Fix this warning by declaring some large data items with

the __far keyword or by compiling with the data=far option.

1.102 scm083

Warning 83: reference beyond object size

Your code used an address beyond the size of the object used as the base for the address calculation. This warning usually occurs when you refer to an element beyond the end of an array, as follows:

```
char c[10];
void myfunc(void)
{
   c[11] = 0;  /* Warning 83 */
}
```

This message can be produced only when the compiler can determine the value of the subscript at compile time, that is, the subscript is a constant.

1.103 scm084

Warning 84: redefinition of pragma or preprocessor symbol "name"
See line number file "filename"

Your code is redefining the preprocessor symbol or #pragma originally defined at the indicated file and line number.

1.104 scm085

Warning 85: return value mismatch for function "name"
Expecting "type1", found "type2"

The expression specifying the value to be returned from a function was not the same type as the function itself. If possible, the value specified is converted to the appropriate type. You can suppress this warning by casting the return expression to the appropriate type.

Some pre-ANSI C code produces many of these warnings when functions declared as int (either explicitly or implicitly) do not return a value. Before the ANSI committee approved the void keyword, declaring functions as returning an int was the correct way to handle functions that returned nothing. You can compile with the nowarnvoidreturn option to suppress these warnings when a function that is declared as returning an int actually returns nothing.

For example, the following two functions generate warning 85:

```
function1(x)
int x;
{
```

```
   }  /* Warning 85 */

   int function2(x)
   int x;
   {
      char *p = NULL;
      return(p);  /* Warning 85 */
   }
```

You can suppress the warning 85 for function1 by compiling with the nowvret option. The nowvret option does not affect function2, which is attempting to return a pointer from a function declared as returning int.

1.105 scm086

```
Warning 86:  formal parameters conflict with prototype
             See line number file "filename"
```

The types of the parameters to the function do not match the types given in the prototype for the function. Check the prototype at the file and line indicated against your definition.

1.106 scm087

```
Warning 87:  argument count incorrect, expecting number arguments
             See line number file "filename"
```

Your code invoked a function with an incorrect number of arguments, according to that function's prototype. Check the prototype at the indicated file and line number against your usage of the function and the actual function definition.

1.107 scm088

```
Warning 88:  argument type incorrect
             Expecting "type1", found "type2"
```

Your code invoked a function with an argument whose type conflicts with the corresponding parameter as declared in the function's prototype. The type of the argument expected is given as type1, and the type of the argument actually provided is type2. If possible, the argument is converted to the appropriate type as if it were cast to that type. If the argument cannot be converted, an error message is produced.

1.108 scm089

```
Warning 89:  constant converted from "type1" to "type2"
```

Your code supplied a constant that conflicted with the expected type. The constant was converted, but the conversion may have caused a loss of precision or lost values.

For example, in the following code, the prototype for the function foo specifies an int, but the function is passed a double constant:

```
void foo(int);

void myfunc(void)
{
   foo(10.67);  /* Warning 89 */
}
```

The double constant is converted to an integer, resulting in an integer argument of 10, and the compiler generates warning 89 to inform you of the conversion.

1.109 scm090

```
Error 90:  invalid argument type specifier
```

An error was made in declaring an argument in a function or prototype declaration.

For example, the following declaration does not specify a type for the parameter y:

```
void foo(int x, y)  /* Error 90 */
{
}
```

1.110 scm091

```
Error 91:  illegal void operand
```

One of the operands in an expression was of type void. The void type represents no value and is, therefore, illegal in most expressions.

1.111 scm092

```
Warning 92:  statement has no effect
```

An expression statement did not produce an assignment, function call, or other action. Such a statement serves no useful purpose and can be eliminated. This warning is often generated when a typographical error has been made in coding the statement.

For example, you might enter the equality (==) operator when you intended to enter the assignment (=) operator:

```
int i;
i == 5;  /* Warning 92 */
```

In this example, the user intended to assign the value 5 to the integer variable i, but because of the extra equals sign, the statement does nothing.

1.112 scm093

Warning 93: no reference to identifier "name"

Your code declared an automatic (local) variable but never used the variable. However, the variable might be used by code that has been excluded from the object module with #if or #ifdef statements. If so, you should enclose your declaration of the local variable in the same #if or #ifdef statement as the code that references the variable.

If you want to suppress warning 93, #pragma msg 93 ignore must be in effect when the compiler reaches the line containing the last closing brace (}) for the function. If you turn the warning off only for the line containing the declaration of the variable, the compiler will still generate message 93 when it reaches the last line of the function. The line number displayed for the message will be the number of the last line in the function.

1.113 scm094

Warning 94: uninitialized auto variable "name"

An automatic (local) variable was used in an expression before it was given a value. Automatic variables are not guaranteed to have any specific value when a function is entered, so an uninitialized automatic variable can create seemingly random bugs. It is possible for this warning to not be produced when appropriate or to be produced incorrectly because the compiler does not check all possible execution paths. In rare cases, the message is produced incorrectly if the variable is used in a loop construct and initialized later in the same loop construct, as shown:

```
void myfunc(void)
{
   int i, j;
   for(i=0; i<10; i++)
   {
      if(i > 0) printf("%d\n", j);  /* Warning 94 */
      j = i;
   }
}
```

Without doing detailed loop analysis, the compiler cannot tell that j is not referenced by the call to printf until after it is initialized. Some cases that use uninitialized variables may not produce the warning.

For example, no warning is produced for the following code even though j is not initialized if x is greater than 5:

```
void myfunc(int x)
{
   int j;

   if(x<5) j = x;

   if(j < 5)  /* j might be uninitialized */
   {
      ...
   }
}
```

1.114 scm095

Warning 95: unrecognized #pragma operand

The keyword following a #pragma statement was not recognized as a SAS/C pragma keyword. Version 6 of the SAS/C Compiler supports the #pragma libcall, amicall, regcall, flibcall, syscall, tagcall, and msg statements.

1.115 scm096

Error 96: missing name for #pragma

In a libcall, flibcall, syscall, or tagcall #pragma statement, the name of the routine to be called was omitted or incorrect.

1.116 scm097

Error 97: bad library base for #pragma

In a libcall, flibcall, or tagcall #pragma statement, the library base was omitted or incorrect.

1.117 scm098

Error 98: invalid data for #pragma

In a libcall, flibcall, syscall, or tagcall #pragma statement, the offset or magic number fields were invalid. In a #pragma msg statement, a syntax error was made specifying the message number or the action to be taken.

1.118 scm099

```
Error 99:  attempt to change a const value
```

Your code attempted to modify a variable declared with the const keyword. Variables declared with the const keyword are read-only and cannot be modified.

1.119 scm100

```
Warning 100:  no prototype declared for function "name"
```

Your code called the named function, but the compiler did not find a prototype for that function. Without a prototype, the compiler cannot perform parameter type checking. Remember, a function that takes no parameters still needs a prototype with void as its parameter list:

```
int func(void);  /* "func" returns "int" and takes no parms */
```

If the function is a SAS/C function, the SAS/C Development System Library Reference tells you which header file to #include to get the correct prototype for the function. If the function is an AmigaDOS function, its prototype is in the include:clib directory in the file for its library. For more information, you can refer to the Amiga ROM Kernel Reference Manual: Libraries, 3rd Edition (Commodore-Amiga, Inc. 1992) or use the grep command to search the header files in the include:clib directory for the prototype.

1.120 scm101

```
Warning 101:  redundant keywords in declaration
```

The same storage class keyword was specified multiple times when declaring a variable, function, or prototype. For example, you may have specified the const or __chip keyword twice.

However, you can specify a typedef with a storage class keyword. Any variables declared with this typedef are automatically assigned to that specified storage class. Therefore, you may get a Warning 101 for specifying the keyword explicitly on a variable declared with the typedef, as in the following example:

```
typedef int __far farint;

farint x;  /* Declares a far integer called "x" */

farint __far y;      /* Error 101 */
```

1.121 scm102

```
Error 102:  conflicting keywords in declaration

You have specified two mutually incompatible storage class keywords when
declaring a variable, function, or prototype.  For example, you may have
specified the __near and __far keywords on a variable definition.

However, you can specify a typedef with a storage class keyword.  Any
variables declared with this typedef are automatically assigned to that
specified storage class.  Therefore, you may get an Error 102 for
specifying a keyword explicitly on a variable which conflicts with the
keyword declared with the typedef, as in the following example:

   typedef int __far farint;

   farint x;  /* Declares a far integer called "x" */

   farint __near y;  /* Error 102 */
```

1.122 scm103

```
Warning 103:  uninitialized constant "name"

The named variable was declared const, indicating that the variable is
read-only, but you have not initialized the variable.  If the variable is
declared static or extern, it is initialized to zero for you but is
probably not useful unless explicitly initialized.
```

1.123 scm104

```
Warning 104:  conversion from pointer to const/volatile to pointer to
non-const/volatile

A pointer to a const or volatile object is being converted to a pointer to
a normal object.  For example, you may be passing the address of a const
variable to a function that takes a normal pointer.  After the pointer has
been converted, the const or volatile attribute is lost.  Therefore, your
const data might get modified, or your volatile data might be kept in a
register.

In the following example, warning 104 is produced when the function foo is
called with the const array data as an argument.

   void foo(char *);

   void myfunc(void)
   {
      const char data[] = "Hello, World!\n";
      foo(data);  /* Warning 104 */
      printf("%s\n", data);
   }
```

```
void foo(char *p)
{
   p[1] = 'a';
}
```

The function foo actually modifies its data by writing to it. Therefore, printf prints the value Hallo, World! instead of Hello, World! because the constant array has been modified.

1.124 scm105

Warning 105: module does not define any externally-known symbols

The C source file and all of its included headers were compiled, but no externally-visible functions or data items were defined. In this case, a valid object file is produced, but it contains nothing that can be accessed, and the file can be removed from your link step if you desire.

1.125 scm106

Error 106: postfix expression not allowed on a constant

Your code attempted to apply a postfix ++ or -- operator to a constant. This message may be generated because of an incorrect macro expansion. You can use the pponly option to generate a preprocessed file and check the macro expansion.

1.126 scm107

Error 107: too many initializers

A data item was being initialized, but too many initializer elements were provided, as in the following example:

```
struct FOO
{
   int i, j;
};

struct FOO foo = {10, 20, 30};  /* Error 107 */
```

1.127 scm108

Warning 108: zero-length arrays are not an ANSI feature

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=108 options. You have declared an array of size

zero, probably as a member of a structure to allow variable-sized
structures. While using zero-length arrays is a common extension to the
ANSI C Standard and is supported by the SAS/C Compiler, the ANSI C Standard
does not allow the use of zero-length arrays.

1.128 scm109

Error 109: invalid use of type name or keyword

A type name (possibly a typedef) has been encountered where it was not
expected. You may have attempted to declare a variable with the same name
as a keyword.

1.129 scm110

Warning 110: enum constant expression is wrong type
              Expecting "type1", found "type2"

An enumeration constant value was explicitly assigned that did not match
the type of the enum being declared. For example, specifying a double
constant when defining an enum or specifying a long constant when the
shortint option is active will generate this message.

Example:

```
   enum COLORS { RED=1.0 }; /* Warning 110 */
```

1.130 scm111

Warning 111: non-portable enum type specified

This message is suppressed by default, but you can enable it with the
STRICT, ANSI, or WARN=111 options. As an extension to the ANSI C Standard,
the SAS/C Compiler supports short, char, and long enum types with the
syntax short enum, char enum, and long enum. To improve the portability of
your code, enable this warning.

1.131 scm112

Warning 112: include file "filename" not in GST

This message is suppressed by default, but you can enable it with the
WARN=112 option. This message lets you know if your code #includes a
header file that is not present in the GST. This option helps you
determine which files should be added to your GST to speed up compilation.

1.132 scm113

```
Error 113:  invalid structure reference
            "<op>" operator is invalid for type "<type>"
```

You have attempted to use the "." structure reference operator with an object that is not a structure, or you have attempted to use the "->" structure pointer dereference operator with an object that is not a pointer. The most common mistake in this area is using "->" on a structure or "." on a pointer to a structure:

```
struct FOO *pFoo;
struct FOO Foo;

Foo->x = 10;  // ERROR!  Foo is a struct, not a pointer!
pFoo.x = 10;  // ERROR!  pFoo is a pointer, not a struct!
```

1.133 scm114

```
Warning 114:  negative shift or shift too big for type
              shifts for type "type" must be between 0 and number bits
```

Your code attempted to shift an integral value by a constant number of bits, but the constant was either negative or higher than the number of bits in the value being shifted, as shown in the following examples:

```
int i = 1;
i = i >> 33;  /* Warning 114 */
i = i << -1;  /* Warning 114 */
```

In the first example, i is 32 bits (an int) but is being shifted 33 bits. Shifting a value by a number of bits higher than the number of bits in the value generates a zero for the result. In the second example, i is being shifted by a negative number of bits. Attempting to shift a value by a negative number of bits usually generates a zero for the result (although this result is undefined by the ANSI C Standard).

1.134 scm115

```
Error 115:  enum constant value "number" out of range for enum type
```

Your code specified a value for an enum constant while defining an enumerated type which is out of range for the type of enum being declared. Remember that enum constants that are not explicitly assigned a numerical value are given the value of the previous constant defined in the same enum incremented by one. This increment might place the constant's value out of the acceptable range, as in the following example:

```
char enum COLORS
{
   RED=254,   /* Numeric value is 254         */
```

```
      GREEN,     /* Numeric value is 255 (254+1) */
      BLUE       /* Numeric value is 256 - too big for char */
   };            /* Error 115 is issued */
```

The enum constant RED is assigned a value of 254. GREEN does not have an explicit value, so it is assigned 254+1, or 255. BLUE does not have an explicit value, so its value would normally be 255+1, or 256. However, the enum is declared as being a char enum, so 256 is too big and error 115 is issued.

1.135 scm116

Warning 116: undefined enum tag "name"

This message is suppressed by default, but you can enable it with the warn=116 option. A reference was made to an enum that has not yet been declared. The compiler may generate this message as a warning if only a pointer to the enum is used, but the compiler generates this message as an error if any other use is made of the enum.

1.136 scm117

Error 117: enum contains no members

An attempt was made to define an enumerated type, but no member names were specified. Use the pponly option to generate a preprocessed file if you have problems eliminating this message.

1.137 scm118

Error 118: conflicting use of enum/struct/union tag "name"

An attempt was made to define an enum, struct, or union that has the same name as a previously defined enum, struct, or union in the same scope. You cannot define an enum with the same name as a struct, union, or another enum.

1.138 scm119

Error 119: identifiers missing from definition of function "name"

A prototype-style function definition was encountered, but the names of the parameters were missing. You can omit the parameter names from a prototype, but the names must be present in the actual function definition.

For example the following code generates error 119:

```
void func(int, int, double);  /* Legal prototype */

void func(int, int, double)   /* Illegal function definition */
{                             /* Error 119 */
}
```

1.139 scm120

```
Warning 120:  Integral type mismatch: possible portability problem
              Expecting "type1", found "type2"
```

This message is suppressed by default, but you can enable it with the strict or warn=120 options. The wrong integral type was passed as a parameter. This error may be a problem on another compiler or another type of computer if the size of an int is different from the size of an int on the Amiga computer. This warning is not a problem if your code needs to compile with the SAS/C Compiler only.

In the following example, the SAS/C Compiler promotes the short integer to a long integer even if you compile with the shortint option:

```
#pragma msg 120 warn
void func(long);

void main(void)
{
   short s = 10;
   func(s);      /* Warning 120 */
}
```

However, pre-ANSI compilers on machines that define the int type to be the same as short may not perform the conversion or may perform the conversion and issue a warning message. In addition, if you remove the prototype, this code will not work on any machine where int is the same as short.

1.140 scm121

```
Warning 121:  hex/octal constant "constant" too large for char
              High bits may be lost
```

Warning 121 is generated if a hexadecimal or octal constant specifies more digits than will fit into a single char. According to the ANSI C Standard, hexadecimal bytes may be specified in a quoted string using the \xhh escape sequence. The ANSI C Standard also states that the escape sequence consumes all valid hexadecimal characters in the string following the \x, even if all the characters consumed will not fit into a single byte, which may not be what you want. For example, in the case of \xabcdefgh, you may want the character 0xab first, followed by the string cdefgh. According to the ANSI C Standard, this escape sequence evaluates to the character 0xef followed by the string gh, because the \x consumes all valid hexadecimal characters. A single byte cannot hold the hexadecimal number 0xabcdef, so the top bits are lost and 0xef is the result. The best way to get the

character 0xab followed by the string cdefgh is to use ANSI string concatenation to terminate the escape sequence:

```
"\xab" "cdefgh".
```

The two strings are concatenated to form the desired sequence.

1.141 scm122

```
Warning 122:  missing ellipsis
```

A function or function prototype was encountered that attempted to specify variable arguments with a trailing comma in the argument list. According to the ANSI C Standard, variable arguments must be specified by a trailing comma followed by an ellipsis (...). The SAS/C Compiler accepts the form without the ellipsis as if the ellipsis had been specified, but other ANSI-conforming compilers may require the ellipsis, as in the following example:

```
void foo(int x, ... );  /* Right */
void bar(int x, );      /* Wrong */
```

1.142 scm123

```
Warning 123:  no tag defined for enumeration
              Cannot construct prototype
```

The genproto option was active, but a prototype could not be generated for a function because an enum was used as a parameter to the function and that enum had no tag, as shown in the following example.

```
void myfunc(colors)
enum {RED, GREEN, BLUE} colors;
{
}
```

The tag is the name normally appearing immediately after the keyword enum in the enum declaration. This message is produced only if the genproto compiler option is used to produce prototypes.

1.143 scm125

```
Error 125:  invalid number
```

The numerical constant specified cannot be represented on the Amiga computer. The constant is probably an integer that is too large or too small (negative) to be represented in four bytes. This error can also be produced if a floating-point number is specified incorrectly or if invalid digits are provided to an octal constant.

The following lines all produce error 125:

```
   double d = 10.3.2;          /* Error 125 */
   int i = 123456789123456789; /* Error 125 */
   int j = 099;                /* Error 125 */
```

In the first line, the value assigned to the floating-point variable has been incorrectly specified. In the second line, the value contains too many digits to fit into an int. In the third line, the value begins with a zero, which makes the value an octal constant, but it contains the digit 9, which is invalid for octal constants.

1.144 scm126

Warning 126: #endif, #else, or #elif out of order

A #endif, #else, or #elif was encountered but no #if or #ifdef was active.

1.145 scm127

Error 127: operand to # operator must be a macro argument

If a preprocessor macro parameter is preceded with the # operator, that parameter is replaced with the literal string consisting of the corresponding argument to the macro. For example, if you pass the argument FOO to a macro, and that argument in the macro definition is preceded by the # operator, that argument expands to the string "FOO". The # operator can be applied only to macro arguments.

1.146 scm128

Error 128: text-from-#error

This message is the result of a #error preprocessor directive in the source code being compiled. The text of the message is taken from the #error line.

1.147 scm129

Error 129: ambiguous structure or union member "name"

Your code referred to a structure or union member that did not exist. In an attempt to resolve the name, the compiler examined the names of all members of substructures or subunions. Two or more members of subaggregates matched the name you provided, so the compiler was unable to determine which member you intended to specify.

```
Suppose you have the following code:

void main(void)
{
   struct FOO
   {
      int x, y, z;
   };

   struct BAR
   {
      int a, b, x;
   };

   struct COMBO
   {
      struct FOO foo;
      struct BAR bar;
   } combo;

   combo.a = 10;  /* Warning 193 */
   combo.x = 10;  /* Error 129 */
}

The reference to combo.a succeeds, although it generates a warning 193
(implicit reference to structure member), because the full reference should
be combo.bar.a.  The reference to combo.x generates error 129, because both
substructures of combo contain a member called x.  The compiler cannot tell
whether you mean combo.foo.x or combo.bar.x, so it issues the error message
129.

See Chapter 11, "Using SAS/C Extensions to the C and C++ Languages," for
more information on implicit structure references.
```

1.148 scm131

```
Error 131:  maximum temporary or formal storage exceeded

The compiler must allocate stack storage to allow you to pass and receive
structures or other parameters to and from functions.

Your function used more stack space than one function is allowed to use.

Parameters are copied onto the stack for each function call, so if you are
passing large structures, your code will run very slowly.  Consider passing
a pointer to your structure instead of the entire structure.
```

1.149 scm132

```
Warning 132:  extra tokens after valid preprocessor directive

The ANSI C Standard does not allow extra text on a preprocessor line, but
```

this extra text may be allowed by some C compilers. For example, many programmers put a descriptive word after a #endif directive indicating the #if to which the #endif belongs. To make your code ANSI-compatible, you should place such descriptive words in comments, as shown in the following example:

```
#ifdef SOMETHING
#endif SOMETHING /* Warning 132 */

#ifdef SOMETHING
#endif /* SOMETHING */  /* No warning */
```

1.150 scm133

```
Error 133:  cannot redefine macro "name"
```

An attempt was made to redefine a built-in preprocessor macro, like __FILE__ or __LINE__. You cannot redefine built-in preprocessor macros.

1.151 scm134

```
Warning 134:  too many arguments
```

A macro definition was encountered with more than the allowable number of arguments. The current ANSI limit for macro arguments is 31, and the SAS/C Compiler does not allow more than 31 arguments. If message 134 is issued, the macro is not successfully defined. Any use of the macro in your code produces a warning for a function with no prototype and possibly a linker error when the name of the macro cannot be resolved by the linker.

1.152 scm135

```
Error 135:  argument count incorrect for macro "name", expecting number
            arguments
            See line number file "filename"
```

Your code invoked a macro with too few or too many arguments. The macro is defined at the specified file and line number.

1.153 scm136

```
Error 136:  invalid use of register keyword
```

Your code used one of the register keywords inappropriately. The register keywords are register, __a0, __d0, __a1, __d1, and so on.

1.154 scm137

```
Warning 137:  ANSI limits #line numbers to between 1 and 32767
```

This message is suppressed by default, but you can enable it with the STRICT, ANSI, or WARN=137 options. The ANSI C Standard states that line numbers specified with a #line directive should be between 1 and 32767. However, the SAS/C Compiler accepts any number that will fit into a signed four-byte integer.

1.155 scm138

```
Error 138:  operation invalid for pointer to void
```

Your code attempted to perform an operation on a void * pointer that is not valid for void *. Such operations include addition, subtraction, array indexing, and dereferencing (unary * operator).

For example, the following code attempts to perform addition on a void * pointer:

```
void *p = NULL;

p = p + 1;  /* Error 138 */
```

1.156 scm139

```
Warning 139:  missing #endif
              See line number file "filename"
```

The compiler encountered an #if, #ifdef, or #elif statement for which there is no matching #endif. The file and line number of the unmatched directive is specified in the message. This message occurs only at the end of the source file, so large portions of your program may have been ignored because of a stray #if or #ifndef with no matching #endif.

1.157 scm140

```
Warning 140:  sizeof operator used on array that has been converted to
pointer
```

Use of the sizeof operator on an array name gives the size of the array. However, if the array name has been converted (cast) to a pointer, the resulting size is the size of the pointer (4 bytes), not the size of the array. Almost any use of an array name in an expression implicitly casts the array name to pointer. To determine the actual size of the array, pass only the array name to the sizeof operator.

In the following example, in the first assignment statement, i gets the

value 4 (the size of a pointer to a character). In the second assignment statement, i gets the value 10.

```
char ary[10];
int i;

i = sizeof(ary+1);  /* Warning 140 */
i = sizeof(ary);
```

1.158 scm142

Error 142: array size never given for "name"

You can declare an array as extern and not specify the first subscript. However, when you define the array, you must specify the subscript. Storage is allocated when you define an array, not when you declare an array as extern. For example:

```
extern char ary1[][10];  /* Legal */

char ary2[][10];         /* Error 142 */
```

1.159 scm143

Error 143: object has no address

An attempt was made to take the address of something that has no address. You may be attempting to take the address of a register variable or an expression.

1.160 scm146

Warning 146: case value out of range for switch type

A case value was specified in a switch statement that can never be taken. For example, if you compile with the shortint option and a switch statement is switching on a short or int value, case values too big to be stored in 16 bits will generate this warning:

```
/* The SHORTINT compiler option is on. */
int i = 10;

switch(i)  /* i is a 16-bit integer */
{
   case 0x08000000:   /* Warning 146 */
      break;

   case 65000:        /* Warning 146 */
      break;
}
```

1.161 scm147

```
Warning 147:  conversion between function and data pointers
```

Your code has cast or otherwise converted a pointer that points to a function to a pointer that points to data. The ANSI C Standard states that this conversion is not legal. This conversion works with the SAS/C Compiler in most circumstances, but you should use the absfuncpointer compiler option if your code is larger than 32K.

1.162 scm148

```
Warning 148:  use of incomplete struct/union/enum tag "name"
              See line number file filename
```

This message is suppressed by default, but you can enable it with the warn=148 option. An incomplete tag is one for which no definition has been seen. The ANSI C Standard allows use of an incomplete tag (for declaring pointers, for example). If you want to know when an incomplete tag is being used, enable this warning. You may also want to enable warning 149.

1.163 scm149

```
Warning 149:  tag "tag-name" was incomplete in prototype
              See line n file "filename"
```

This message is suppressed by default, but you can enable it with the warn=149 option. An incomplete tag was discovered in a prototype and the function is defined in the same module. An incomplete tag is one for which no definition has been seen. The ANSI C Standard requires the incomplete tag's scope to end at the end of the prototype. Therefore, any definition of the function later, even if the tag in question has been defined, may generate an error message and terminate the compilation. You may want to turn on warning 148 to get more information about incomplete tags. If you do not want to see all the incomplete tag information, you can identify most problems by enabling only warning 149.

In Version 6.0, this warning was produced any time a prototype contained an incomplete tag, regardless of whether the function was defined later in the same module.

The following function definition produces an error on many ANSI-conforming compilers because the structure FOO referred to in the definition is considered to be a different structure FOO than the one referred to in the prototype.

```
void func(struct FOO *);

struct FOO
{
   int x, y, z;
};
```

```
/* Many compilers issue an error here. */
void func(struct FOO *foo)
{
}
```

This error could be fixed by moving the definition for the structure FOO before the prototype.

1.164 scm150

Warning 150: the keyword "name" is meaningless for itemtype

When declaring a function or data item, you have used a keyword that is meaningless for that type of item. For example, you cannot specify the following:

```
void __chip foo(int);  /* Warning 150 */
```

It is meaningless to say that a function is to be allocated in __chip memory.

The __near and __far storage class keywords are valid on both functions and data items, but have slightly different meanings. See the descriptions of the data=near and code=near options in Chapter 8, "Compiling and Linking Your Program," for more information.

Some keywords that are valid on functions are also valid on data items because the data items may be pointers to functions of the designated type. For example, you may have a function that returns a pointer to a __regargs function. Keywords in this category include __stdargs, __regargs, __asm, and __interrupt.

Other keywords need to be present only on the function definition. You do not need to add them to function pointers and function prototypes. Keywords in this category include __stackext and __saveds. If these keywords appear on a data item, warning 150 is generated and the extra keyword is ignored. For more information on specifying keywords, see Chapter 11, "Using SAS/C Extensions to the C and C++ Languages."

1.165 scm152

Error 152: cannot define function via typedef name

Your code attempted to define a function using a typedef, as shown below:

```
typedef int foo(int);

foo bar  /* Error 152 */
{
}
```

According to the ANSI C Standard, you cannot define a function using a typedef. The SAS/C Compiler does not accept such a definition.

1.166 scm154

Warning 154: no prototype declared for function pointer

Function pointers require prototypes just like functions. To declare a prototype for a function pointer, enter the parameter list instead of the empty parentheses in the definition, as shown in the following example:

```
void (*func1)();  /* Warning 154 */

void (*func2)(int, double, long);  /* No warning */
```

Remember that a pointer to a function that takes no parameters still requires a prototype:

```
void (*func3)(void);  /* Function pointer takes no parms */
```

1.167 scm155

Warning 155: no statement after label

The C language does not allow a label immediately before the curly brace ending a block (}). If you enter a label in this position, the SAS/C Compiler generates this warning. Enter a semicolon (representing a NULL statement) after the label to suppress the warning, as shown in the following example:

```
void func1(void)
{
   goto foobar;
      /* more code */
   foobar:    /* Warning 155 */
}

void func2(void)
{
   goto foobar;
       /* more code */
   foobar: ;   /* No warning */
}
```

1.168 scm156

Warning 156: operation with/comparison of types "type-1" and "type-2"

Your code attempted to use pointers to two different types in an operation or comparison. Examine your code carefully; if it seems correct, cast one

of the pointers to the other pointer's type.

1.169 scm158

Error 158: invalid type name

The compiler expected a type name for a cast or offsetof operation, but the provided name was not recognized as a valid type name. Check the preprocessed output to make sure you are providing the correct information.

1.170 scm159

Warning 159: use of unary minus on unsigned value

This message is suppressed by default, but you can enable it with the strict or warn=159 options. Your code has used the unary minus operator (-) on an unsigned variable. Using this operator on an unsigned variable may not produce the expected result because the result is still a non-negative number.

1.171 scm161

Warning 161: no prototype declared at definition for function "name"

An old-style function definition was encountered, and no prototype was in scope. The prototype must appear in the C file or a header file before the definition of the function, or the function definition itself must be a prototype-style definition.

1.172 scm162

Warning 162: non-ANSI use of ellipsis punctuator

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=162 options. You have declared a function with the ellipsis punctuator (...) but the function has no arguments. The ANSI C Standard requires functions that take a variable number of arguments to take at least one fixed argument.

1.173 scm163

Warning 163: initialization of auto struct, union, or array

This message is suppressed by default, but you can enable it with the strict or warn=163 options. Your code has initialized an automatic

structure, union, or array. The ANSI C Standard allows you to initialize automatic structures, union, and arrays, but many pre-ANSI compilers do not.

1.174 scm164

Warning 164: & applied to array

This message is suppressed by default, but you can enable it with the strict or warn=164 options. Your code has used the address (&) operator on an array name. You should instead take the address of the first element in the array, or use the array name without the address operator.

For example:

```
int ary[10];
int *iptr;

iptr = &ary;       /* warning 165 */
iptr = ary;        /* OK          */
iptr = &ary[0];    /* OK          */
```

1.175 scm165

Warning 165: use of narrow type in prototype

This message is suppressed by default, but you can enable it with the warn=165 option. This warning is provided for detecting situations that may cause problems on other compilers if your code mixes function prototypes and old-style function definitions. See Chapter 13, "Writing Portable Code," for information on using narrow types in function declarations. See also the descriptions of messages 176 and 179.

1.176 scm166

Error 166: unrecoverable error or too many errors
Terminating compilation

Your program has exceeded the default or specified maxerr or maxwarn values, or an error has occurred that prevents the compiler from producing meaningful results.

The default maximum number of errors is 50. By default, any number of warnings may be generated.

1.177 scm169

```
Warning 169:  incompatible operands of conditional operator (?:)
              "type1" conflicts with "type2"
```

The operands of the ?: conditional operator must be of compatible types. Your code supplied types to the ?: operator that were not compatible.

For example, the following code generates warning 169:

```
int func(void)
{
   int i = 0;
   struct FOO *foo = NULL;

   return (int)(i > 0 ? foo : i);  /* Warning 169 */
}
```

The expression following the question mark (?) is of type struct FOO *. The expression following the colon (:) is of type int.

This message can be an error if it is not possible to convert the types in question. This can occur if one of the types is a structure or union.

1.178 scm170

```
Warning 170:  overflow during operation on constants
```

A constant expression overflowed the limits of the type in which it was being calculated. Perhaps you have added or multiplied two large integers, thereby resulting in a number too large to represent in a four-byte integer.

1.179 scm176

```
Warning 176:  implicitly promoted formal "name" conflicts with prototype
              See line number file "filename"
```

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=176 options. You are defining a function using an old-style definition, which means that any narrow types (char, short, and float) in your definition are implicitly widened to their non-narrow equivalents (int, int, and double, respectively). However, a prototype is in scope that gives the narrow version of the type.

Functions that call your function will not know that your function is using an old-style definition and, with some compilers, may pass an incorrect value. An incorrect value is never passed using the SAS/C Compiler on the Amiga hardware. See Chapter 13, "Writing Portable Code," for information on using narrow types in function declarations. See also the descriptions of messages 165 and 179.

1.180 scm178

```
Warning 178:  indirect call without indirection operator
```

This message is suppressed by default, but you can enable it with the strict or warn=178 options. You called a function using a function pointer, but did not use the indirection (*) operator to dereference the pointer first. The SAS/C Compiler generated correct code, but on pre-ANSI compilers this code may not work.

Example:

```
   void (*funcptr)(int);

   funcptr(10);  /* Warning 178 */
   (*funcptr)(10);  /* No warning */
```

1.181 scm179

```
Warning 179:  narrow type used in old-style definition
```

This message is suppressed by default, but you can enable it with the strict or warn=179 options. Your code has used a narrow type (char, short, or float) in an old-style definition. See Chapter 13, "Writing Portable Code," for information on using narrow types in function declarations. See also the descriptions of messages 165 and 176.

1.182 scm180

```
Warning 180:  no space between macro name and its replacement list
```

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=180 options. Your code is defining a preprocessor macro that takes arguments but does not have at least one blank after the closing parentheses of the argument list. The ANSI C Standard requires white space after the closing parentheses.

For example:

```
   #pragma msg 180 warn
   #define ADD(a,b)(a+b)  /* Warning 180 */
```

1.183 scm181

```
Warning 181:  "name" was declared both static and external
              See line number file "filename"
```

Your code has declared a function or a global variable as both static and external at different places. Both the function and its prototype must agree on whether the function is static.

1.184 scm182

```
Warning 182:  static function "name" declared but not defined
              See line number file "filename"
```

Your code had a prototype for a function declared static but never defined the function. The function definition may be hidden with #if or #ifdef statements.

1.185 scm183

```
Warning 183:  inline function "function" declared but not defined
              See line number file "filename"
```

Your code had a prototype for a function declared __inline but never defined the function. The function definition may be hidden with #if or #ifdef statements.

1.186 scm184

```
Warning 184:  unterminated character constant
```

Your code has specified a multibyte character constant (such as 'ab'), and you did not compile your code with the mbchar option; or, if you compiled your code with the mbchar option, the multibyte character is larger than the permitted four bytes.

1.187 scm185

```
Error 185:  comma expected
```

The compiler expected a comma but did not find one. This error may be produced because of errors in previous statements. Fix all previous errors before fixing this one.

1.188 scm186

```
Warning 186:  implicit conversion between pointer and scalar
```

Your code has converted a pointer to an integer while doing static initialization. You can suppress this warning by casting the pointer to the appropriate type.

1.189 scm187

```
Warning 187:  negative value assigned to unsigned type

This message is suppressed by default, but you can enable it with the
strict or warn=187 options.  Your code has assigned a negative constant to
an unsigned variable.  In doing so, your code is in effect assigning a very
large positive number to the variable, which may or may not be the action
you intended.
```

1.190 scm188

```
Error 188:  Function and data definitions not allowed when creating a GST

NEW WITH VERSION 6.55

You enabled the MAKEGST option to create a Global Symbol Table, but your
header files attempt to define a function or variable.  You cannot have
function or data definitions in a header file that you want to add to a
GST.

A definition differs from a declaration in that it actually allocates
storage.  For example, the following are definitions:

   int x;
   void func(void)
   {
      return;
   }
   __inline int add(int x, int y)
   {
      return x + y;
   }

but these are only declarations:

   extern int x;
   void func(void);

You can have only one definition per function or data item, but you
can have as many declarations as you like.  ANSI C says that if you
declare an item and include an initializer, the declaration becomes
a definition:

   extern int x = 10;

In this case, the "extern" keyword is ignored because an initializer
is present.

You can often remove definitions from your header files by replacing
them with declarations, then moving the definition to one of your
C or C++ source files.

For more information on GSTs, please see page 29 of your Library
```

Reference Manual, Version 6.50.

1.191 scm189

Warning 189: <option> option differs from the one used to build the GST

NEW WITH VERSION 6.55

You are trying to use a Global Symbol Table, but your default or specified options to the SC command do not match the ones used to build the GST.

Certain compiler options are very fundamental and must be the same between the creator and the user of a GST. These options include the MATH= option, the SHORTINT option, and the UNSIGNEDCHAR option. You may need to recreate your GST if you have modified one of these options.

The compiler produces this message as an error if the values used for the SHORTINT option do not match between the compilation and the GST, or if either the compilation or the GST used MATH=FFP and the other uses any other math option (or no math option). It is produced as a warning if the MATH= options do not match exactly or if the UNSIGNEDCHAR option does not match. If produced as a warning, the message is suppressed by default, but may be enabled with the WARN=189 or STRICT compiler options.

1.192 scm190

Warning 190: #include ignored because header already included
See line number file "filename"

This message is suppressed by default, but you can enable it with the warn=190 option. The nomultipleincludes compiler option is active, and your code included the same header file more than once. The compiler ignores the additional #include statement. You can use this message to locate all such multiple includes if you want to modify your header files to not include the same file more than once.

You can get even more information about which header files were included by using the listincludes compiler option.

1.193 scm192

Error 192: wrong size for enum

An enum variable was declared to be of a different type than the base enum was when it was defined. Suppose you have the following code:

```
char enum COLORS = {RED, GREEN, BLUE};
```

```
   enum COLORS color;  /* Warning 192 */
   char enum COLORS color2;      /* correct */
```

The enum variable color should be declared the same size as the base enum type, which is char enum. The base enum type overrides the enum variable's type definition.

1.194 scm193

```
Warning 193:  implicit reference to struct/union member
              Reference assumed to be "reference"
```

Your code has referred to a struct or union member name that is actually a member of a substructure or subunion of the original. The compiler has searched all substructures and subunions and determined that exactly one member matches the name you specified, so it is using that member. The reference printed in the message text is the fully expanded name of the member that it is using. If you intend to take advantage of this feature of the SAS/C Compiler, you may disable this warning with the ignore=193 option. If you choose to disable this warning, your code may not work on other compilers.

For more information on using implicit structure references, see Chapter 11, "Using SAS/C Extensions to the C and C++ Languages."

1.195 scm194

```
Warning 194:  too much local data for NEAR reference,
              some changed to FAR
```

You have declared more than 32K of static data. The compiler can address up to this amount using 16-bit offsets. Amounts greater than 32K must be addressed using 32-bit offsets. You may have compiled with the data=faronly option and declared some data with the __near keyword. If your entire project is in one source file or is compiled with data=faronly, you can ignore this warning unless you get an error later in the compilation or link. Otherwise, you must eliminate some near data in one of four ways:

- Compile your program with the strmerge compiler option. This option moves all string constants to the code section, thereby moving them out of the near data section.
- Declare read-only data as static const. When you compile with the stringmerge option, this data will also be moved to the code section.
- Add the __far keyword to some of your larger external or static data items to move those items from the near section to the far section.
- Specify the data=far compiler option to move all data items except those declared with the __near keyword to the far section.

NOTE: The second line of message 194 is not printed unless you compiled

with the data=auto option, or you compiled with the data=faronly option and declared data with the __near keyword.

1.196 scm195

Warning 195: nested comment detected

This message is suppressed by default, but you can enable it with the warn=195 option. The compiler found the start of a comment (/*) inside of a comment. Some compilers allow comment nesting and others ignore the start of the second comment. The ANSI C Standard does not allow nested comments. If you choose to use nested comments, you should specify the comnest compiler option. However, using the comnest option prevents your code from running on most ANSI-compliant compilers.

1.197 scm196

Warning 196: specified include directory not found: "name"

The named directory does not exist or could not be accessed.

1.198 scm198

Warning 198: __regargs and __asm cannot be used on a varargs function

Functions that take variable numbers of parameters always pass their parameters on the stack. The __regargs and __asm keywords are invalid for these functions.

1.199 scm199

Error 199: unbalanced comment
See line number file filename

Your code has a comment open (/*) with no corresponding comment close (*/). This condition is not detected until the end of the C source file.

1.200 scm200

Warning 200: No register specified for parameter to __asm function

You have declared a function with the __asm keyword, but you did not select a register for one or more of the parameters.

For example:

```
int __asm foo(int x);  /* WRONG!  No register for 'x'! */

int __asm bar(register __d0 int x);  /* RIGHT! */
```

1.201 scm202

```
Warning 202:  relational comparison between unsigned and zero
```

Your code compared an unsigned value with zero in such a way that the expression will always be true or always be false.

For example:

```
unsigned int x;
if(x < 0)  /* Never true! */
...

if(sizeof(x) >= 0)  /* Always true! */
...
```

This message is suppressed by default, but you can enable it with the strict or warn=202 option.

1.202 scm204

```
Warning 204:  invocation of macro "macro-name" too large or not terminated
```

Your code invoked a macro but did not supply enough closing parentheses to terminate the invocation. Also, check any of the macros that are invoked by the macro that you invoked directly in your code.

1.203 scm209

```
Warning 209:  macro name invocation may have multiple side effects
```

You have passed an expression with a side effect as an argument to a macro, and that macro has evaluated the argument more than once. Operators with side effects are ++, --, +=, /=, *=, -=, >>=, <<=, |=, &=, and =.

For example, the max and min macros evaluate each argument twice. If you invoke the max macro as MAX(i++, j), the first argument is evaluated twice, and two post-increments take place.

A typical definition of the max macro is as follows:

```
#define MAX(x,y) ((x) > (y) ? x : y)

int i = 0;
```

```
   i = MAX(i++, 0);   /* Warning 209 */
```

The macro expands to:

```
   i = ((i++) > (0) ? i++ : 0);
```

The expansion contains the expression i++ twice, so if that branch of the ?: operator is executed, the variable i is incremented twice.

This warning may sometimes be produced incorrectly, but you should examine each case to determine whether there is an error.

Function calls are also considered side effects, but function calls produce warning 217 instead of warning 209.

1.204 scm212

```
Warning 212:  item "name" already declared
              See line number file "filename"
```

This message is suppressed by default, but you can enable it with the warn=212 option. You have declared an item twice. For example, you may have specified the prototype for a function twice, or you may have declared an extern twice. The declarations do not conflict, or the compiler would generate error 72, but code is harder to maintain if it defines the same item in more than one place. You may want to use the nomultipleincludes option to suppress multiple #includes of the same file if you enable warning 212.

1.205 scm213

```
Warning 213:  empty argument to preprocessor macro "macro-name"
```

This message is suppressed by default, but you can enable it with the strict, ansi, or warn=213 options. Your code calls a preprocessor macro with no argument text. This action is accepted by the SAS/C Compiler but is not permitted by the ANSI Standard.

For example:

```
   #define FOO(a,b)   a

   FOO(10,)
```

1.206 scm216

```
Warning 216:  symbol "name" found
```

You have compiled with the fsym compiler option, and the compiler has found a definition of one of the symbols that you specified as the parameter to

the fsym option.

1.207 scm217

Warning 217: macro invocation may call function multiple times

You have passed a function call as an argument to a macro that evaluates its arguments more than once. This action may cause incorrect results.

See also the description of message 209.

1.208 scm218

Error 218: declaration found in statement block

The compiler found a declaration where it expected a statement. You may have a statement in the middle of your declaration block, or vice-versa. Any declarations found after this error are added as external variables, which suppresses warnings about undefined variables but may create additional errors later if your code declares a variable of the same name. In the following example, the variable y is declared after the first statement of the function:

```
void func(void)
{
   int x;
   x = 10;
   int y;   /* Error 218 */
}
```

1.209 scm220

Warning 220: old-fashioned assignment operator
taken as "operators"

This message is suppressed by default, but you can enable it with the strict or warn=220 options. Older compilers allowed the operators =-, =+, and so on. In ANSI C, these operators are specified as -=, +=, and so on. The older specification is ambiguous when assigning certain expressions (for example, x=-5; and x= -5;). To resolve this ambiguity, newer compilers use -=, and some pre-ANSI compilers use =-. Therefore, you should enter at least one space between the equals (=) sign and whatever operator follows it to ensure portability.

1.210 scm223

```
Error 223:  "filename" is not a valid GST file

The specified filename was supplied using the gst compiler option as a
Global Symbol Table file, but it is not a valid GST file.  Delete the bad
file and try re-creating it.
```

1.211 scm224

```
Warning 224:  item "name" already defined
              See line number file "filename"

This message indicates that a variable, function, or typedef is being
defined for the second time in a given scope.  This message is normally an
error, but if the redefinition is harmless, the compiler allows it and
issues the message as a warning instead.  As shown in the following
example, the compiler allows redefinition of typedef names in the same
scope if the new type is identical to the old type.  The compiler does not
allow redefinition of functions or variables in the same scope.

   typedef int foo;
   typedef int foo;  /* Warning 224 */

   void func(void)
   {
   }

   void func(void)   /* Error 224  */
   {
   }
```

1.212 scm225

```
Warning 225:  pointer type mismatch
              "type1" does not match "type2"

Your code has performed an assignment or some other operation on two
incompatible different pointer types or on a pointer type and an arithmetic
type.

For example:

   struct FOO
   {
      int x, y, z;
   } foo;
   int *ip;

   ip = &foo;  /* Warning 225 */
```

1.213 scm226

```
Error 226:  cannot convert "type1" to "type2"
```

The compiler was unable to perform a requested or implicit conversion. For example, if one of the types is an instance of a structure, the structure cannot be cast to another structure type or to an arithmetic type.

For example:

```
struct FOO
{
   int x, y, z;
} foo;
int i;

i = (int) foo;  /* Error 226 */
```

1.214 scm301

```
Warning 301:  Indirect reference through NULL pointer.
```

The program being optimized contained code to dereference a pointer with a NULL value. The results of such a dereference are undefined but may include an addressing exception or a system crash.

1.215 scm302

```
Warning 302:  Type punning involves representation change symbol
```

A variable defined with one type was used with a different type. This practice is known as type punning. This message can occur, for example, when an assignment is made to a member of a union and then the value is accessed from a different member of the union.

1.216 scm303

```
Warning 303:  Reference has overlapped definition symbol
```

The indicated identifier has at least two definitions that partially overlap its storage.

1.217 scm304

```
Warning 304:  Dead assignment eliminated symbol
```

symbol was assigned a value on the indicated line, but the value was never used. The assignment was eliminated.

1.218 scm305

```
Warning 305:  Uninitialized variable symbol
```

The named identifier was not initialized before it was used. The identifier's value is unpredictable.

1.219 scm306

```
Note 306:  reason function inlined: function name {from line n}
```

The named function was expanded inline at the line associated with the message. The reason is __inline, complexity = c, or local, depending on which inline option caused the inlining to occur. c is the complexity of the function (not the value of the complexity option). The section {from line n} is only included when the location at which the call was expanded differed from the location of the call. This message happens when levels of inlining are occurring. The value n is the line number of the call.

1.220 scm307

```
Warning 307: return value missing in inline function
```

Your code is defining an inline function, but the function does not return the required type. A common cause of this is omitting the "void" return type from functions that return nothing:

```
__inline func(int a, int b)
{
   ...code...
}
```

If the function really returns nothing, add the "void" keyword:

```
void __inline func(int a, int b)
{
    ...code...
}
```

If it should return a value, make sure that all paths return values:

```
int __inline func(int a, int b)
{
   if(a>b)
      return a-b;
   /* Note: This path does not return a value */
}
```

1.221 scm308

```
Note 308: inline function does not use formal parameter <symbol>

You have defined an inline function that does not refer to the
specified formal parameter.  This may be acceptable programming
practice; if it is, turn off this warning with a #pragma msg or
the NWARN 308 option.  To turn off the message for just this
function, surround the function definition with

#pragma msg 308 ignore push

   <function definition>

#pragma msg 308 pop
```

1.222 scm402

```
Error 402:  Wrong number of parms to builtin function

The wrong number of parameters were passed to a builtin function.  The
builtin function will not be used.
```

1.223 scm403

```
Error 403:  Argument(s) to function-name must be int type

The builtin function requires an integer parameter.
```

1.224 scm404

```
Error 404:  __builtin_fpc requires MATH=68881 option

The __builtin_fpc function was used without the math=68881 option.
```

1.225 scm405

```
Error 405:  Floating point opcode must be a constant.

The parameter to __builtin_fpc must be a constant, since it is the opcode
for a 68881 instruction.
```

1.226 scm406

```
Error 406:  Offset from library base must be a constant

A #pragma libcall statement was declared with a variable for the offset for
the jump vector.
```

1.227 scm407

```
Error 407:  Offset from library base must be negative
```

Jump vectors are always negative from the library base.

1.228 scm408

```
Error 408:  Insufficient parameters for library call
```

The number of parameters to a #pragma libcall does not match the magic number in the #pragma statement.

1.229 scm409

```
Error 409:  Too many parameters for library call
```

The number of parameters to a #pragma libcall does not match the magic number in the #pragma statement.

1.230 scm410

```
Error 410:  Invalid register specification for getreg/putreg
```

The register number is out of range. Refer to the header file dos.h for a list of registers.

1.231 scm411

```
Error 411:  Value for getreg/putreg must be an integral type
```

The register must be a constant.

1.232 scm412

```
Error 412:  FP register used without co-processor
```

You must specify the math=68881 option to use floating-point registers in #pragma flibcall statements.

1.233 scm415

```
Error 415:  Same register used twice for parameters
```

The same register was specified for more than one parameter in an __asm function.

1.234 scm416

```
Error 416:  No register specified for ASM call
```

A parameter was specified without a register in an __asm function.

1.235 scm417

```
Error 417:  No Data register available to reach far formals.
            Reduce the size of auto variables or reduce the
            number of register parameters.
```

If you have more than 32K of auto variables and formal parameters, the compiler needs an extra register to reach the formal parameters. If all registers are used as parameters, you will get this message.

1.236 scm1101

```
Error 1101:  Illegal token.
```

A symbol that is not recognized as a valid token has been detected in the input. For example, you may have a symbol that you intended to be a C++ identifier, but it contains a character not permitted in identifiers. For instance, symbols cannot have a number as the first character of the identifier.

1.237 scm1102

```
Error 1102:  Can't find file:  filename.
```

The file specified in a #include directive cannot be found.

1.238 scm1103

Error 1103: Invalid file name.

The filename in a #include directive must be enclosed in double quotes ("") or angle brackets (<>). See "F.3.13 Preprocessing Directives" in Appendix 3, "Implementation Defined Behavior," for information on when to use quotes or brackets.

1.239 scm1104

Error 1104: End of file encountered in comment.

The end of the source file has been detected while a comment was being processed (and before the comment terminator was seen).

1.240 scm1105

Warning 1105: Invalid escape sequence.

An invalid escape sequence has been detected.
An invalid escape sequence is a backslash (\) followed by a character not valid in such a context. For example, \z is not a valid escape sequence.

1.241 scm1106

Error 1106: illegal preprocessor directive.

A # character followed by a symbol not recognized as a valid preprocessor directive has been detected. You may have misspelled the directive.

1.242 scm1107

Warning 1107: Extra token(s) after preprocessor directive.

A #if, #else, or #endif directive has been followed by some text not within a comment.

1.243 scm1108

Error 1108: Missing identifier in preprocessor context.

An identifier is required in the given preprocessor context but is not present. For example, if your program says #if defined instead of #if defined myfunc, this message is issued.

1.244 scm1109

Error 1109: Redefinition of preprocessor symbol: symbol.

A previously defined symbol has been redefined. This message is produced as a warning if the new definition is the same as the old definition, but it is produced as an error if the two definitions are different.

1.245 scm1110

Error 1110: Missing ')' in macro call.

A prior occurrence of a left parenthesis has been detected in a macro call but has not been matched by a right parenthesis.

1.246 scm1111

Error 1111: Missing argument to preprocessor macro.

A preprocessor macro has been called with fewer arguments than the macro requires. Check the definition of the macro to determine which arguments are missing.

1.247 scm1112

Error 1112: Missing comma in preprocessor expression.

A comma is required to separate the arguments in the definition or call of a function-like preprocessor macro.

1.248 scm1113

Error 1113: Illegal operator in preprocessor context.

The preprocessor has detected an operator that it cannot understand. Examples of operators that are illegal in the preprocessor context are: ++, --, ., and ->.

1.249 scm1114

Error 1114: Missing operand in preprocessor context.

An operand was expected but not found.

1.250 scm1115

```
Error 1115:  Illegal expression in preprocessor context.
```

The preprocessor has encountered an expression that it cannot understand. For example, an operand may be expected but is not present, or the operand is of a non-integral type (for example, a floating-point constant).

1.251 scm1116

```
Error 1116:  Preprocessor number not a true number.
```

A token beginning with a digit resembles an integer or floating-point constant, but is not correct. For example, you may have mistyped a digit in a hexadecimal constant or mistyped a floating-point exponent field.

1.252 scm1117

```
Error 1117:  Integer required in preprocessor expression.
```

A non-integer operand has been detected in a preprocessor expression. The logical operators (!, ||, and &&) require integral operands.

1.253 scm1118

```
Error 1118:  Extra #else or #elif.
```

A #else or #elif directive has been found without a matching #if directive.

1.254 scm1119

```
Error 1119:  Extra #endif.
```

A #endif directive has been found without a matching #if directive.

1.255 scm1121

```
Error 1121:  Invalid #line format.
```

The preprocessor has encountered an invalid #line directive. The form of the #line directive is

```
        #line number "string"
```

number must be an integer constant and string must be enclosed in double quotes.

1.256 scm1122

```
Error 1122:  Missing parenthesis in preprocessor expression.
```

A beginning left parenthesis has been detected in an expression but was not matched by a right parenthesis.

1.257 scm1123

```
Error 1123:  Illegal use of # operator.
```

The identifier following the # operator is not an argument to the macro being defined.

1.258 scm1124

```
Error 1124:  #error directive.
```

The preprocessor has encountered a #error directive. This message is issued whenever a #error directive is encountered.

1.259 scm1125

```
Error 1125:  Illegal ## expression.
```

Either the first or second operand of the ## operator is missing. This may happen if the ## expression begins or ends the line.

1.260 scm1126

```
Error 1126:  Illegal operand of ## operator.
```

An operand of the ## expression is not an identifier. You may have misspelled the identifier.

1.261 scm1127

```
Error 1127:  Unterminated string or character constant.
```

A string (starting with a double quote) or a character constant (starting with a single quote) was begun but not terminated.

1.262 scm1129

```
Error 1129:  Character literals must contain at least one character.
```

Two consecutive single quotes were found with no intervening characters. Character literals must have at least one character between the single quotes.

1.263 scm1130

```
Error 1130:  Unterminated preprocessor conditional.
```

The end of the source file was reached while a conditional preprocessor directive (such as the #elif directive) was pending. One or more #endif directives are missing.

1.264 scm1200

```
Error 1200:  Syntax error more-explanation.
```

A syntax error has been detected. There are many forms of message number 1200. The amount of information provided in these messages is dependent upon the context in which the error occurs.

1.265 scm1205

```
Error 1205:  Newline within string or character literal.
```

The newline character (indicating an end of line) has been found within a string literal or character literal. Usually this occurs when a terminating single quote or double quote has been omitted from the literal. To cause a newline character to appear in the output, use the \n escape sequence, as in "Hi\n".

1.266 scm1206

```
Error 1206:  Bad character in input (hex-number).
```

The specified character has been detected in the input and is not an acceptable character for a token. Normally this condition occurs only if your source file has had odd characters inserted in it, perhaps as a result of uploading or downloading the file from one machine to another.

1.267 scm1208

```
Warning 1208:  C style comment starting on line line-number never ends.

There is no terminating */ sequence to the comment.  Either you have
forgotten the comment terminator or you intended the comment to be a C++
comment.
```

1.268 scm1319

```
Error 1319:  'identifier' not declared.

The specified identifier is not declared.  It may be misspelled.
```

1.269 scm1320

```
Error 1320:  No such class: 'identifier'.

The specified identifier is not a class, but is used in a place where a
class name is required, such as before the ::  operator or in a
base-specifier-list.
```

1.270 scm1321

```
Error 1321:  'struct' or 'class' used on 'enum identifier'.

The specified identifier is an enum tag, but the keyword struct or class
was used instead of enum.
```

1.271 scm1322

```
Error 1322:  'enum' used on 'class identifier'.

The specified identifier is a class tag, but the keyword enum was used
instead of struct or class.
```

1.272 scm1323

```
Error 1323:  'identifier' previously declared to be a type-name.

The specified identifier was previously declared to have some other type.
```

1.273 scm1324

Error 1324: 'identifier' redefined.

The specified identifier was previously defined in this scope.

1.274 scm1325

Error 1325: Scoped declaration in parameter list.

The :: operator cannot be used in parameter lists.

1.275 scm1326

Error 1326: Label 'label-name' not defined.

The specified label appeared as the target of at least one goto statement
in the previous function, but the label was never defined in the function.
Labels are defined in a function using the label: statement notation.

1.276 scm1327

Error 1327: Label 'label-name' previously defined.

The specified label was defined more than once in the same function.

1.277 scm1328

Error 1328: Repeated keyword or type name: 'keyword'.

A keyword or type name was used more than once within a single declaration.

1.278 scm1329

Error 1329: Conflicting keywords or type names: 'keyword-1' and
'keyword-2'.

An illegal combination of keywords or type names was used in this
declaration.

1.279 scm1330

Error 1330: Must be integral, pointer, or member pointer.

An expression of a non-testable type was used where a testable value is required. Testable types are all the integral, pointer and member-pointer types. Testable values are required as the first expression of a ?: operator and as the test for the if, while, do while, and for statements.

1.280 scm1331

Error 1331: Must be integral.

An expression of a non-integral type was used where an integral valued expression is required. An integral value is required in a case label and in the test expression of a switch statement.

1.281 scm1332

Error 1332: No such conversion.

An illegal conversion was specified in a cast operator.

1.282 scm1334

Error 1334: Expression is not modifiable.

You have attempted to assign a value to an expression that cannot be modified. An lvalue is any expression that can appear on the left side of an assignment operation.

For example, the result of a cast is not an lvalue; therefore, the following statement is invalid:

```
long x;
(short)x = 2;  /* Error 1334 */
```

The following examples also generate this error message:

```
#define ADDONE(x) (x)++
ADDONE(12);     /* Cannot increment a constant */
ADDONE(&g);     /* Cannot assign to an address */

if(func(10)=j-2);  /* "==" was intended, not "=" */

&x = &y;        /* Cannot assign to an address */
```

1.283 scm1335

Error 1335: Invalid use of '&' address-of operator (object).

The address-of operator (&) was applied to an object that was not addressable. Examples of nonaddressable objects are bitfields and register variables. Also, you cannot take the address of an overloaded function except as an initializer.

1.284 scm1336

Error 1336: Cannot initialize (variable) with (initializer).

The initializer is of a type that cannot be converted to a type required by the variable it is initializing.

1.285 scm1337

Error 1337: Preprocessor error.

The preprocessor has encountered an error that is beyond its capabilities to diagnose.

1.286 scm1338

Error 1338: Unexpected end of file.

The parser reached the end of the input source before it expected to. This may result during error recovery from a previous syntax error in the input source. Otherwise, it usually indicates that a closing brace (}) or semicolon (;) has been omitted at the end of your source.

1.287 scm1339

Warning 1339: A non-lvalue array was converted to a pointer.

Only arrays that are lvalues can be converted to pointers. (See message 1334 for an explanation of lvalue.) Because the array is not converted to a pointer, operator [] should not be applied to it because operator [] requires a pointer. Also the array should not be assigned to a pointer variable. If you want only to access the array, you may ignore this warning. However, if you want to alter the value of an array element, you should treat this message as an error.

1.288 scm1340

Error 1340: The base name 'class-1' is ambiguous in class 'class-2'.

The class class-1 occurs more than once as a base of the second class, class-2.

1.289 scm1342

Error 1342: Conversion from a virtual base class ('class-name') to a derived class is not allowed.

Virtual base classes cannot be converted, either explicitly or implicitly, to derived classes.

1.290 scm1343

Error 1343: Ambiguous conversion to integral type from 'class class-name'.

The class has defined multiple conversions to an integral type.

1.291 scm1344

Error 1344: Ambiguous conversion to pointer from 'class class-name'.

The class has defined multiple conversions to pointer.

1.292 scm1345

Error 1345: Ambiguous conversion to testable from 'class class-name'.

The class has defined multiple conversions to one or more of arithmetic, pointer, or member-pointer types.

1.293 scm1346

Error 1346: Ambiguous conversion to derived member pointer.

An ambiguous reference to the derived member-pointer has been found. Resolve the ambiguity by qualifying the pointer name with its class name.

1.294 scm1347

Error 1347: Ambiguous conversion of overloaded function pointer.

An ambiguous reference to the overloaded function pointer has been found. Resolve the ambiguity by qualifying the function pointer name with its class name.

1.295 scm1348

Error 1348: Ambiguous conversion to class.

An ambiguous reference to the class to which conversion is being made has been found. For example, you may have a file-scope function that has the same name as a class (and hence, the same name as the conversion function of that class).

1.296 scm1349

Error 1349: Ambiguous conversion.

An ambiguous conversion has been specified. For example, a constructor and a function cannot have the same name.

1.297 scm1350

Error 1350: Ambiguous function call.

The function name used in the call is ambiguous. For example, two classes may both have a function of the same name and a class name has not been used to qualify the function name in the call.

1.298 scm1351

Error 1351: Overloaded functions ('function-1' and 'function-2') used ambiguously in conditional expression.

Two overloaded functions were used as the second and third operands to a conditional operator (?:). These overloaded functions have more than one function type in common. Cast one or both operands to the desired function pointer or member function pointer type.

1.299 scm1352

Error 1352: Ambiguous common base class: class-name.

The reference to the base class is ambiguous. Resolve the ambiguity by further qualifying each occurrence of this name.

1.300 scm1353

Error 1353: Ambiguous member name: member-name.

The expression used to refer to the member refers to more than one function, object, type, or enumerator. Resolve the ambiguity by qualifying the member name with its class name.

1.301 scm1354

Error 1354: Non-static member 'member-name' must be used with dot, arrow, or address-of operator.

Non-static members must be used only in the following contexts:

- after a dot or arrow operator
- as a member-pointer (as in &class-name::member-name)
- within a sizeof or offsetof expression
- within a non-static member function of a class that contains or inherits the member (where the this-> operation is implied).

1.302 scm1355

Error 1355: Value of an undefined class cannot be used.

You have tried to dereference a pointer to an undefined class. A solution is to include the definition of the class before it is used.

1.303 scm1356

Error 1356: An array may not be the target of an assignment.

Array assignment (that is, assignment of an array name) is not supported in C or C++.

1.304 scm1357

```
Error 1357:  A function may not be the target of an assignment.

Function assignment (that is, assignment to a function name) is not
supported in C or C++.
```

1.305 scm1358

```
Error 1358:  Cannot operation a pointer to type.

You cannot add to or subtract from a pointer to void, a function pointer,
or a pointer to an undefined class.  Nor can you use the indirection
operator (*) on a pointer to void or a pointer to an undefined class.  The
error message explains which of these mistakes you have made.
```

1.306 scm1359

```
Error 1359:  Typedef names cannot be declared in parameter lists.

A typedef name has been encountered in a parameter list.  Move the
definition of the typedef name outside the parameter list (that is, to file
scope).
```

1.307 scm1361

```
Error 1361:  Cannot take the address of a member of virtual base class.

The & operator cannot be applied to a member of a virtual base class.
```

1.308 scm1362

```
Error 1362:  Invalid initializer.

The initializer is of a type that cannot be converted to a type required by
the variable it is initializing.
```

1.309 scm1363

```
Error 1363:  Invalid use of void.

Only functions or pointers can be declared void.
```

1.310 scm1364

Error 1364: Cast to undefined class not allowed.

The class to which a cast is made must be previously defined.

1.311 scm1365

Error 1365: Cannot find offset into non-class.

Only members of classes can have offsets.

1.312 scm1366

Error 1366: Cannot find offset into undefined class.

A class must be defined before the offset of one of its members can be taken.

1.313 scm1367

Error 1367: Invalid use of the scope operator.

The scope operator (::) can be used only in such expressions as C3::mem or C1::C2::C3::mem where C1is a class in which class C2 is declared, C3 is a class declared in C2, and mem is a member of C3.

1.314 scm1368

Error 1368: Cannot find the offset of 'object'.

It is illegal to take the offset of a member function, a static member or a bitfield member. In particular, because the number of bits in a bitfield may be less than the number of bits in a char, or its number of bits may not comprise an integral number of chars, a bitfield in C++ has no size.

1.315 scm1369

Error 1369: Cannot find offset because class 'class-name' has no member named 'member-name'.

The second argument to the offset operation must be a member of the class specified in the first argument.

1.316 scm1370

Error 1370: Cannot take the size of an undefined class.

A class must be defined before its size can be determined.

1.317 scm1371

Error 1371: Cannot dereference pointer to undefined class.

A class must be defined before a pointer to an object of its type can be dereferenced.

1.318 scm1372

Error 1372: No such constructor.

The constructor referred to does not exist.

1.319 scm1373

Error 1373: 'identifier' previously declared as type-1. Cannot be defined as type-2.

A name declared using the union specifier cannot be defined using struct or class. Similarly, a name declared as struct or class cannot be defined using union.

1.320 scm1374

Error 1374: No such member 'member-name'.

The identifier referred to is not a class member.

1.321 scm1375

Error 1375: Member 'member-name' redeclared.

The specified member of the class has been declared more than once.

1.322 scm1376

Error 1376: 'identifier' not a definable member.

Only functions and static data members can be defined outside the class declaration.

1.323 scm1377

Error 1377: 'this' may occur only in a (non-static) member function.

You cannot use the this keyword outside the context of a non-static member function.

1.324 scm1378

Error 1378: Cannot create a new value of a function.

The new operator cannot be applied to a function type. Functions cannot be allocated by means of the new operator, although function pointers can.

1.325 scm1379

Error 1379: Cannot create a new value of a reference.

The new operator cannot be applied to a reference type. Because a reference type is not an object, a pointer to it could not be returned by operator new.

1.326 scm1380

Error 1380: Cannot create a new instance of an undefined class.

A class must be fully defined before the new operator can be used to create a new instance of the class.

1.327 scm1382

Error 1382: Missing array size in expression.

One or more dimensions of the given array have not been specified.

1.328 scm1383

Error 1383: Class 'class-name' has no default constructor.

You cannot use operator new to allocate an array of class objects if the class does not have a default constructor.

1.329 scm1384

Error 1384: Cannot initialize new array.

An array created by means of the new operator cannot be initialized by specifying a brace-enclosed initializer list.

1.330 scm1385

Error 1385: Cannot delete an object of an undefined class.

The delete operator cannot be applied to an object whose class has not been defined.

1.331 scm1386

Error 1386: Length expression of array must be integral.

The size of an array cannot be specified as a floating-point number.

1.332 scm1387

Error 1387: No match for call to function or overloaded operator.

The argument list given for a function call did not match any of the possible parameter lists.

1.333 scm1388

Error 1388: Missing constructor body.

A constructor declaration was followed by a colon (:), but no constructor body ({) was found after the colon.

1.334 scm1389

Error 1389: Non-virtual functions ('function-name') cannot be declared pure.

Only virtual functions can be declared pure. For more information, refer to section r.10.3 in The C++ Programming Language, Second Edition.

1.335 scm1391

Error 1391: Uninitialized const identifier.

A const identifier must have an explicit initialization.

1.336 scm1392

Error 1392: Uninitialized const identifier or reference: identifier.

A const identifier must be initialized explicitly. The declaration of a reference must contain an explicit initializer unless one of the following is true:

- the extern specifier has been used
- the reference declaration is a class member declaration within a class declaration
- the reference declaration is a declaration of a function parameter or function return type.

1.337 scm1393

Error 1393: Const identifier or reference member 'member-name' must be initialized.

A const identifier or reference must be initialized explicitly. This message is caused by a constructor for a class with a const or reference member where the const or reference member is not initialized with a mem-initializer in the constructor. For more information, refer to section r.12.6.2 in The C++ Programming Language, Second Edition.

1.338 scm1394

Error 1394: Member 'member-name' must have initializer, class 'class-name' has no default constructor.

If a class has a constructor (but does not have a default constructor), objects of that class must be initialized. A member is initialized by including a mem-initializer for it on each constructor for the class

containing the member. For example, if the type of X::a is a class with a constructor (but no default constructor) you receive this message if you omit the a(10) in the following code:

```
X::X() : a(10), b(11)

{ . . . }
```

1.339 scm1395

Error 1395: Base 'class-name' must have initializer, class 'class-name' has no default constructor.

If a class has a constructor (but does not have a default constructor), it must be initialized. A base class is initialized by including a mem-initializer for it on each constructor for the derived class.

For example, if b is a base class of X and a class with a constructor (but no default constructor), you receive this message if you omit the b(11) in the following code:

```
X::X() : a(10), b(11)

{ . . . }
```

1.340 scm1396

Error 1396: Virtual base class 'class-name' must have initializer since class has no default constructor.

If a class has a constructor (but does not have a default constructor) it must be initialized. A virtual base class is initialized by including a mem-initializer for it on each constructor for the derived class. The example in the explanation of Error 1395 is applicable in the virtual base class case as well.

1.341 scm1397

Error 1397: 'identifier' is not a base class or member of class 'class-name'.

The specified identifier is not a member or base class of the class being constructed.

1.342 scm1398

Error 1398: Member access through protected base class not allowed for 'member-name'.

The specified member cannot be accessed because it has been inherited from a protected base class, and the function or initializer using the member is not a friend or member of a class derived from that base.

1.343 scm1399

Error 1399: Member access through private base class not allowed for 'member-name'.

The specified member cannot be accessed because it has been inherited from a private base class, and the function or initializer using the member is not a friend or member of the class that is derived directly from that base.

1.344 scm1400

Error 1400: Base access through protected base class not allowed.

The expression that is attempting to access the base class is not in a function or initializer that has access to it. Because the base class is protected, only functions of the following types have access to the base class:

- members or friends of the class that declared the base class
- members or friends of classes derived from the class declaring the base class.

Initializers for members of a class have the same access privileges as functions of that class. For more information, refer to section r.11.2 in The C++ Programming Language, Second Edition.

1.345 scm1401

Error 1401: Base access through private base class not allowed.

The expression that is attempting to access the base class is not in a function or initializer that has access to it. Because the base class is private, only functions that are members or friends of the class that declared the base class have access to the base class. Initializers for members of a class have the same access privileges as functions of that class. For more information, refer to section r.11.2 in The C++ Programming Language, Second Edition.

1.346 scm1402

Error 1402: Cannot access protected member 'member-name'.

The expression that is attempting to access the member mem is not in a function or initializer that has access to it. Because mem is protected, only functions of the following types have access to mem:

- members or friends of the class that declared mem
- members or friends of classes derived from the class declaring mem.

Initializers for members of a class have the same access privileges as functions of that class. For more information, refer to section r.11 in The C++ Programming Language, Second Edition.

1.347 scm1403

Error 1403: Cannot access private member 'member-name'.

The expression that is attempting to access the member mem is not in a function or initializer that has access to it. Because mem is private, only functions that are members or friends of the class that declared mem have access to mem. Initializers for members of a class have the same access privileges as functions of that class. For more information, refer to section r.11 in The C++ Programming Language, Second Edition.

1.348 scm1404

Error 1404: Virtual function 'function-name' declared in virtual base 'class-name' must be overridden.

The virtual function is declared in a virtual base class. It is also overridden in at least two classes derived from the base class and inherited by the class that caused this message. The specified virtual function must be declared in the class that caused this message.

1.349 scm1406

Error 1406: Parameter of type 'void'.

Parameters to functions cannot be declared to be of type void.

1.350 scm1407

Error 1407: Default argument expression missing.

If a formal parameter has a default argument value, then all the parameters after this one must also have default argument values.

1.351 scm1408

Error 1408: Multiple declarations of function specifying default arguments.

The default argument for a formal parameter can be given in only one function declaration.

1.352 scm1409

Error 1409: Arrays cannot contain elements of type 'void'.

Array elements must be of some type other than void.

1.353 scm1410

Error 1410: Arrays cannot contain bitfields.

Array elements must be of some type other than bitfield.

1.354 scm1411

Error 1411: Arrays cannot contain functions.

You can define an array of function pointers, but not an array of functions.

1.355 scm1412

Error 1412: Functions cannot return functions.

A function can return a function pointer, but it cannot return a function.

1.356 scm1413

Error 1413: Functions cannot return arrays.

A function can return a pointer, but it cannot return an array.

1.357 scm1414

Error 1414: Functions cannot return bitfields.

Bitfields cannot be returned by functions.

1.358 scm1415

Error 1415: Functions cannot return undefined classes.

If a function is intended to return a class, that class must first be defined.

1.359 scm1416

Error 1416: Pointers cannot point to references.

References are not addressable and so cannot be referenced by pointers.

1.360 scm1417

Error 1417: Pointers cannot point to bitfields.

Bitfields are not addressable and so cannot be referenced by pointers.

1.361 scm1418

Error 1418: References cannot refer to references.

A reference whose value is a reference is not permitted.

1.362 scm1419

Error 1419: References cannot refer to bitfields.

A reference cannot refer to a bitfield.

1.363 scm1420

Error 1420: References cannot refer to objects of type 'void'.

There are no void objects, so there cannot be a reference to one.

1.364 scm1421

Error 1421: Member pointers cannot point to bitfields.

Bitfields cannot be referenced by member-pointers.

1.365 scm1422

Error 1422: Member pointers cannot point to references.

A reference cannot be referred to by a member-pointer.

1.366 scm1423

Error 1423: Member pointers cannot point to objects of type 'void'.

Because there are no void objects, a pointer cannot point to one.

1.367 scm1424

Error 1424: Bitfields must be of integral type.

Bitfields cannot be of floating-point type.

1.368 scm1425

Error 1425: Overloaded functions with indistinguishable arguments.

Two functions have the same name and the same parameter list. Either delete one of the functions, or change its definition.

1.369 scm1426

Warning 1426: K&R C style function definition.

A Kernighan and Ritchie (K&R) C style function definition has been encountered. C++ requires function definitions to be of prototypical form, but K&R style function definitions are permitted as an extension. This message is only to warn you about the use of this extension. If you do not want to see this warning, use the ignore compiler option to turn it off.

1.370 scm1427

Error 1427: K&R C style functions cannot return classes with constructors or destructors.

If a function is to return a class with a constructor and destructor, it must be defined by means of a prototypical function definition, as opposed to using a Kernighan and Ritchie (K&R) C style function definition.

1.371 scm1428

Error 1428: Conversion function must be a member function.

A conversion function for a class must be defined as a member function.

1.372 scm1429

Error 1429: Destructor function must be a member function.

The destructor for a class must be defined as a member function.

1.373 scm1430

Error 1430: Conversion function 'function-name' not correctly declared.

A return type was specified for a conversion function, or formal parameters were given for a conversion function. Declarations of conversion functions do not specify the return type in the usual way. The return type is part of the name of the function. Also, conversion functions cannot take arguments.

1.374 scm1431

Error 1431: Destructor function 'destructor' not correctly declared.

A destructor has been declared to be something other than a function, or a return type was specified for a destructor. Destructors must be functions and declarations of destructors cannot specify a return type (not even void).

1.375 scm1432

Error 1432: Copy constructor for a class may not take an argument whose type is that class.

Copy constructors cannot take an argument whose type is the class of which the copy constructor is a member. Typically, you can copy objects of class ABC by declaring a copy constructor of the form ABC::ABC(const ABC&). For more information, refer to section r.12.1 in The C++ Programming Language, Second Edition.

1.376 scm1433

Error 1433: Operator function 'function-name' not correctly declared.

An operator function must be either a member function or have at least one parameter of type class.

1.377 scm1434

Error 1434: Invalid linkage specifier.

The extern keyword must be followed by a string literal containing either "C" or "C++". This error may also be caused by an extra or misspelled token after an extern keyword. "C" and "C++" must be specified in uppercase.

1.378 scm1435

Error 1435: Linkage differs from prior declaration.

The linkage specified is not the same as that specified in a prior declaration of the function.

1.379 scm1436

Error 1436: Unknown linkage convention.

SAS/C C++ implements only the "C" and "C++" linkage conventions. Linkage to other languages must be specified using the SAS/C language keywords.

1.380 scm1437

Error 1437: Missing class name.

A class name was expected but not found. This may be caused by a misplaced comma.

1.381 scm1438

Error 1438: Repeated base class.

A base class can be mentioned only once in the base-list. For more information, refer to section r.10.1 in The C++ Programming Language, Second Edition.

1.382 scm1439

Error 1439: Objects of abstract classes ('object-name') cannot be declared.

Classes that have pure virtual functions (abstract classes) can be used only as base classes; they cannot be used to declare variables or members of other classes. It is permissible to declare pointers to abstract classes.

1.383 scm1440

Error 1440: Object of type 'void'.

Only a function can be declared to be of type void. A pointer can be of type void *.

1.384 scm1441

Error 1441: Static members ('member-name') of a local class may not be initialized.

This is one of the limitations of local classes. All static data members of local classes are automatically initialized to zero.

1.385 scm1442

Error 1442: Cannot use undefined enum 'identifier'.

Enumerations must be defined before they can be used.

1.386 scm1443

Error 1443: Enum constants ('identifier') must be initialized with integral values.

The values of an enumeration must be integral values (rather than, for example, floating-point values).

1.387 scm1444

Error 1444: A class cannot be a member of itself.

No member of a class can be of the same type as that class (although a class can have as a member a pointer or reference to that type of class).

1.388 scm1445

Error 1445: Cannot declare members of an undefined class.

A class must be defined before any of its members can be declared.

1.389 scm1446

Error 1446: Cannot declare arrays of an undefined class.

If an array of instances of a given class is to be declared, the class must first be defined.

1.390 scm1447

Error 1447: Cannot declare variables of an undefined class.

Classes must be defined before objects of that class can be declared.

1.391 scm1448

Error 1448: Cannot initialize data members in member declaration.

Data members cannot be initialized within the class definition. Non-static data members must be initialized in the mem-initializer of each constructor function; for more information, refer to section r.9.4 in The C++ Programming Language, Second Edition. static data members must be initialized outside the class; for more information, refer to section r.12.6.2 in The C++ Programming Language, Second Edition.

1.392 scm1449

Error 1449: Member function of a local class must be defined within that class: class-name.

A member function of a local class must be defined within that class and not merely declared within the class.

1.393 scm1450

Error 1450: Member 'member-name' declared 'void'.

A member cannot be of type void.

1.394 scm1451

Error 1451: 'friend' used on non-function.

The friend keyword has meaning only in function declarations inside a class.

1.395 scm1452

Error 1452: 'friend' can only be used inside a class.

The friend keyword has meaning only in function declarations inside a class.

1.396 scm1453

Error 1453: Invalid syntax for access declaration.

A member declaration contains inconsistent information. Friend function declarations require the friend keyword. Access declarations cannot specify any type information. Member declarations inside the class definition cannot specify the class using the scope operator. Either add the friend keyword, remove the type information, or remove the class name and scope operator, depending upon which type of declaration you intended.

1.397 scm1454

Error 1454: Invalid access adjustment: 'member-name'.

Access to a base class member cannot be adjusted in a derived class that defines a member of the same name.

1.398 scm1455

Error 1455: Access cannot be changed, but only reinstated.

Access declarations must declare the inherited member to have the same access as the member in the class from which it is inherited.

1.399 scm1456

Error 1456: Previously declared as a member in this class.

An access declaration cannot specify a name defined in the derived class.

1.400 scm1457

Error 1457: 'class::member' is not a member of a base class.

The specified member is not a member of any base class.

1.401 scm1458

Error 1458: Access declaration names class that is not a base of this class.

You have tried to adjust the access to a member of a base class by using an access declaration, but the base class name you have used is not truly a base class of the derived class. You may have misspelled the base class name.

1.402 scm1460

Error 1460: Constructor function 'constructor' not correctly declared.

An incorrectly declared constructor has been found. Perhaps you have specified a return type for a constructor, or the name used in the declaration of the constructor is not the same as the name of the class. Declarations of constructors cannot specify a return type (not even void) and the name of the constructor must be the same as the class name.

1.403 scm1461

Error 1461: Destructor function 'destructor' not correctly declared.

An incorrectly declared destructor has been found. Perhaps you have declared it to be something other than a function, or specified a return type for the destructor. Destructors must be functions and declarations of destructors cannot specify a return type (not even void). The name of the destructor must be a tilde (~) followed by the class name.

1.404 scm1462

Error 1462: Operator function 'function-name' not correctly declared.

An incorrectly declared operator function has been found. Perhaps you have declared it with the wrong number of arguments. Unary operators take only one argument and binary operators take two arguments. Also, member functions have an implicit this argument, which counts against this limit. So, for example, a unary operator declared as a member function has no explicit formal parameters.

1.405 scm1463

Error 1463: Static functions ('function-name') cannot be virtual.

static functions cannot be virtual. Remove the virtual keyword from the declaration of the static function.

1.406 scm1464

Error 1464: Constructors ('constructor') cannot be virtual.

Constructor functions cannot be virtual. Remove the virtual keyword from the declaration of the constructor function.

1.407 scm1465

Error 1465: Static functions ('function-name') cannot be used to override virtual functions.

A static member function was declared to have the same name and argument types as a virtual function inherited from a base class. Use a different name for the static function.

1.408 scm1467

Error 1467: Linkage specification cannot be used in a member declaration ('member-name').

Linkage declarations can be used only in file-scope declarations of non-members.

1.409 scm1468

Error 1468: Cannot define classes or enums in return types or parameter lists.

Classes and enumerations cannot be defined in parameter lists (prototypes) or in return type declarations.

1.410 scm1469

Error 1469: Invalid parameter name 'parameter'.

Parameter names cannot be operator function names, operator conversion names, or destructor names.

1.411 scm1472

Error 1472: Formal 'argument' is not listed in function declaration.

All arguments to a function must be listed in the parameter list of the function.

1.412 scm1473

Error 1473: Initialized local extern 'variable'.

External variables declared inside a function cannot be initialized in the function. Instead, they must be initialized by another declaration outside the function.

1.413 scm1474

Error 1474: Type names (name) cannot be initialized.

Declarations of typedef names cannot contain initializers.

1.414 scm1475

Error 1475: Class with constructors must have an initializer.

Variables declared to be of classes that have constructors must be initialized.

1.415 scm1476

Error 1476: Cannot define classes or enums in type names.

Classes and enumerations cannot be defined in cast operators, new operators, sizeof expressions, or offsetof expressions.

1.416 scm1477

Error 1477: Not a function.

A file-scope declaration followed by a mem-initializer or a function body must declare a function.

1.417 scm1478

Error 1478: A mem-initializer may be used only within constructor functions.

This message occurs when a colon follows a function declarator, but the function is not a constructor.

1.418 scm1479

Error 1479: Base or member 'identifier' re-initialized.

The same base class or member was initialized by two mem-initializers in the same constructor function. For example, the following code generates this message because the a(10) part appears twice:

```
X::X() : a(10), a(10)

{ . . . }
```

1.419 scm1480

Error 1480: Old style base initializer cannot be used on class with no bases.

An old style base initializer is a mem-initializer with no specified name. They can be used only for classes with a single base class. For example, if X has no base classes, the following code tries to initialize a non-existent base and generates this message:

```
X::X() : (10)

{ . . . }
```

To correct the error, delete the (10) part. For more information, refer to section r.18.3.2 in The C++ Programming Language, Second Edition.

1.420 scm1481

Error 1481: Old style base initializer cannot be used on class with multiple base classes.

An old style base initializer is a mem-initializer with no specified name. They can be used only for classes with a single base class. For example, if class X has more than one base, the following code is ambiguous and generates this message:

```
X::X() : (10)
```

```
   { . . . }

To correct the error, insert the name of a specific base class before the
(10).  For more information, refer to section r.18.3.2 in The C++
Programming Language, Second Edition.
```

1.421 scm1482

```
Warning 1482:  Statement is unreachable.

The statement flagged will never be executed.  You may want to check your
program logic.
```

1.422 scm1483

```
Error 1483:  'case' label must be within a switch statement.

case labels are not allowed outside of a switch statement.
```

1.423 scm1484

```
Error 1484:  'default' label must be within a switch statement.

The default label is not allowed outside of a switch statement.
```

1.424 scm1485

```
Error 1485:  'continue' must be within a loop ('do', 'for', or 'while')
statement.

The continue statement is not allowed outside of a loop statement.
```

1.425 scm1486

```
Error 1486:  'break' must be within a switch or loop ('do', 'for', or
'while') statement.

The break statement is valid only within switch or loop statements.
```

1.426 scm1487

```
Error 1487:  Missing return value.

The return value of the function was not specified.
```

1.427 scm1489

Error 1489: Return value given for constructor, destructor, or void function.

A return value is not allowed in a constructor, destructor, or void function.

1.428 scm1490

Error 1490: Missing function name in function declaration.

A function name was expected but not found. You may have misspelled the name of a constructor function.

1.429 scm1491

Error 1491: Illegal formal declaration list in prototype function definition.

A function definition included both a prototype and one or more Kernighan and Ritchie (K&R) C style argument declarations. These two styles cannot be mixed in a single definition.

1.430 scm1492

Error 1492: Formal ('argument') must be declared in function header identifier list.

In Kernighan and Ritchie (K&R) C style function definitions, formals declared in the formal declaration list must have already been specified in the identifier list of the function.

1.431 scm1493

Error 1493: Expression in array declarator must be constant expression.

Array declarations that specify a size must specify it as a non-negative integral constant expression.

1.432 scm1494

Error 1494: Expression in array declarator must be integral.

Array declarations that specify a size must specify it as a non-negative integral constant expression.

1.433 scm1495

Error 1495: Expression in array declarator must be positive.

Array declarations that specify a size must specify it as a non-negative integral constant expression.

1.434 scm1498

Error 1498: Invalid bitfield size.

Bitfield sizes must be a non-negative constant expression less than or equal to the size of the specified allocation unit.

1.435 scm1499

Error 1499: Cannot use undefined class 'class-name' as base class.

The specified class must be defined before it can be used as a base class for another class.

1.436 scm1500

Error 1500: Missing declaration-specifier.

Declaration specifiers are required in all non-function declarations. They are also required in function declarations in parameter lists.

1.437 scm1501

Error 1501: Illegal use of 'item' in local member function.

Local member functions can use type names, static variables, extern variables and functions, and enumeration constants only from the enclosing scope.

1.438 scm1502

Error 1502: A class cannot be derived from a union ('union-name').

A class cannot be derived from a union.

1.439 scm1503

Error 1503: A union ('union-name') cannot be derived from another class.

A union cannot be derived from another class.

1.440 scm1504

Error 1504: Constant expression contains a division by zero (0).

Constant expressions cannot contain division by zero.

1.441 scm1506

Error 1506: Cannot take the size of a function.

Functions have no size, although function pointers do.

1.442 scm1507

Error 1507: Cannot take the size of a bitfield.

Bitfield sizes are not expressible in bytes.

1.443 scm1508

Error 1508: Cannot take the size of void.

Because there are no void objects, you cannot take the size of one.

1.444 scm1509

Error 1509: Cannot take the size of array with unspecified length.

The array was declared without giving its length, so its size cannot be determined.

1.445 scm1510

Warning 1510: Cannot jump into a block to a label after a declaration having an initializer.

It is prohibited to jump into a block (using a goto statement) if the destination label occurs after a declaration in the same block that has an initializer. Should such a jump occur, the object in question would not be initialized but could be referenced in subsequent code.

1.446 scm1511

Error 1511: Overloaded member functions ('function-name') may not be both static and non-static.

All member functions of the same name must be either static or non-static.

1.447 scm1512

Error 1512: Function hides a virtual function from base class.

Because the function hides a virtual function, the virtual function is not called.

1.448 scm1513

Error 1513: Overriding virtual function has different return type.

An overriding virtual function cannot change the return type.

1.449 scm1514

Error 1514: Arrays cannot contain references.

Arrays of references are not allowed.

1.450 scm1515

Error 1515: Previous declaration of function had different return type.

An earlier declaration of the function specified a different return type.

1.451 scm1516

Error 1516: Cannot have two extern "C" functions with same name ('name').

In a program, only one of a set of overloaded functions of a given name can be declared extern "C".

1.452 scm1517

Error 1517: Previous declaration differed in the use of __builtin.

All declarations of a function must be consistent in the use of the __builtin keyword. All must have it, or none may have it.

1.453 scm1518

Error 1518: object-type ('expression') cannot be used in default argument expressions.

Non-static members, formal parameters, and automatic variables cannot be used in default argument expressions.

1.454 scm1522

Error 1522: Keyword can only be used on functions.

A keyword that can be used only in function declarations has been used in a declaration of some other type. For example, virtual and inline can be used only in function declarations.

1.455 scm1523

Warning 1523: 'keyword' cannot be applied to object-type.

It is invalid or meaningless to apply the keyword in the specified context. For example, it is meaningless to apply a storage class keyword such as auto to the definition of a class (although auto can be applied to the definition of an object whose type is that class). Depending upon the combination of keywords in question, this may be treated as a warning or an error.

1.456 scm1524

Error 1524: Previous declaration was not static.

A function was first declared non-static and later declared static.

1.457 scm1525

Error 1525: Function declared 'inline' after first use.

A function was used prior to its declaration as an inline function.

1.458 scm1528

Error 1528: Member functions must be C++ functions.

Only C++ functions (and not functions in C or in other languages) can occur as member functions.

1.459 scm1530

Error 1530: Previous errors prevent continuation.

The source program has one or more errors that prevent the compilation from continuing.

1.460 scm1531

Error 1531: A declaration must declare something.

An empty declaration has been encountered. This message usually is caused by the omission of a variable name in a declaration (leaving only a sequence of keywords or type symbols).

1.461 scm1532

Error 1532: function-name cannot have 'storage-type' storage class.

A member function declared const or volatile cannot also be declared static.

1.462 scm1533

Warning 1533: Extra comma at end of enumeration list.

An enumeration list has an extraneous comma after its last member.

1.463 scm1534

Warning 1534: Enum value: value is used for both 'enum-1' and 'enum-2'.

An enumeration value has been repeated. This warning is given if two enum constants in the same enumeration type have the same value. This may be what you intended; the warning is given in case it is not intended.

1.464 scm1535

Error 1535: Cannot overload 'main'.

main can not be overloaded.

1.465 scm1536

Error 1536: Cannot call or take the address of 'main'.

The function main cannot be called, nor can its address be taken.

1.466 scm1537

Error 1537: 'main' cannot be 'storage-type'.

The function main cannot be declared static or inline.

1.467 scm1538

Error 1538: Anonymous classes cannot have constructors or destructors.

An anonymous (unnamed) class cannot have a constructor or a destructor.

1.468 scm1539

Error 1539: Destructor names ('destructor') must be the same as their class name ('class').

The tilde (~) can be used only to declare a destructor if that destructor has the same name as the class for which it is a destructor.

1.469 scm1540

Error 1540: Expression in array declarator must not be negative.

In C++, an array must be declared to be of a positive size. As an extension, SAS/C C++ allows the declaration of zero-length arrays.

1.470 scm1541

Error 1541: Cannot allocate array of class 'class-name' with no default constructor.

In order to allocate an array of a class with the new operator, the class must have a default constructor.

1.471 scm1542

Error 1542: Invalid constructor given for member 'member-name'.

An invalid constructor has been encountered. This happens in two circumstances:

- in constructors that do not give an explicit member initializer for a member that is an array of classes without a default constructor
- in constructors that give a multi-argument member initializer for a non-class member.

1.472 scm1543

Error 1543: 'operand-1' and 'operand-2' are not compatible types for conditional operator.

The two operands are not of compatible type for use with the conditional operator.

1.473 scm1544

Error 1544: (type-1) operator (type-2): Invalid type for binary operator.

One of the types displayed is inappropriate for the operator in question or is incompatible with the other type in this context.

1.474 scm1545

Error 1545: 'operand' is of invalid type for postfix operator 'operator'.

The postfix operator (++ or --) cannot be applied to an object of the given type.

1.475 scm1546

Warning 1546: 'operator' is invalid for operand type 'operand'.

The operator cannot be applied to an operand of this type. Depending upon the operator/operand pair in question, this may be either a warning or an error.

1.476 scm1547

Error 1547: 'object' is of invalid type for call operator.

The call operator, (), cannot be applied to an object of the given type.

1.477 scm1548

Error 1548: Invalid pointer conversion from 'type-1' to 'type-2'.

It is not possible to convert a pointer to an object of the given type.

1.478 scm1549

Warning 1549: Non-const and/or non-volatile member function called with const and/or volatile object.

It is an error to call a non-const member function for a const object or a non-volatile member function for a volatile object, but because many other compilers fail to diagnose this error, SAS/C C++ treats it as a warning. Ignoring this warning allows non-const functions to change data declared as const.

1.479 scm1550

Warning 1550: Non-constant reference 'reference-object' initialized with a non-lvalue.

The reference in question is to a non-const but has been initialized with something that is not an lvalue. See Error 25 for a description of lvalues.

1.480 scm1551

Error 1551: Cannot take size of pointer to overloaded function 'function-name'.

Because an overloaded function name does not uniquely determine the function designated, a pointer to such a function likewise is not uniquely determined and consequently its size may be indeterminate as well. Accordingly, sizeof cannot be applied to such a pointer.

1.481 scm1553

Error 1553: Error writing to output file: filename.

An error has been encountered in writing to the output file. This could be caused by a variety of environmental factors (such as a lack of disk space).

1.482 scm1554

Error 1554: Inline member function does not end.

The end of the input file has been encountered before the closing brace (}) was seen for a member function.

1.483 scm1555

Error 1555: Static function 'function-name' was not defined.

The specified function was declared static but has not been defined in this source file.

1.484 scm1556

Error 1556: Global anonymous unions must be static.

An anonymous union declared at file scope must be declared as static.

1.485 scm1557

Error 1557: Anonymous unions may not have function members.

You cannot define an anonymous union that contains function members. If you want function members, use another construct, such as a plain union, struct, or class.

1.486 scm1558

Error 1558: Anonymous unions may not have private or protected members.

You cannot define an anonymous union that contains private or protected members. If you want private or protected members, use another construct, such as a plain union, struct, or class.

1.487 scm1559

Error 1559: 'identifier' redeclared in anonymous union.

The specified identifier was previously declared to have some other type.

1.488 scm1560

Error 1560: An anonymous union cannot be declared as a static member.

An anonymous union has no name for linkage to the definition.

1.489 scm1562

Error 1562: Conflicting declaration of name 'identifier' reserved for purpose.

You have inadvertently used a name that the compiler reserves for its own uses. Choose another name.

1.490 scm1564

Error 1564: Cannot initialize a function ('function-name').

You cannot initialize a function. You can initialize only variables.

1.491 scm1565

Error 1565: Static members (member-name) cannot be initialized by a mem-initializer.

Static members should be initialized by the definition of the static member outside the class. For example, this message is issued if you use the following code and a is a static member:

```
X::X() : a(10)

{ . . . }
```

1.492 scm1566

```
Error 1566:  Enum constants (identifier) cannot be initialized by a
mem-initializer.

Enum constants should be initialized inside the enum declaration, as they
are in C.

For example, this message is issued if a is an enumeration constant:

   X::X() : a(10)

   { . . . }
```

1.493 scm1567

```
Error 1567:  Types must match in a delete expression:  type-1->~type-2.

When calling a destructor for a built-in type, it is required that the two
types specified in the call be the same.
```

1.494 scm1568

```
Error 1568:  Cannot create a new value of a void.

The new operator cannot be applied to void.  Because void is not an object
type, a pointer to it could not be returned by operator new.
```

1.495 scm1569

```
Error 1569:  Loop in -> operators.

The pointed-to expression uses a user-defined operator -> that either
returns the class that contains the operator, or returns a class that
contains another operator -> which in turn returns the class containing the
original operator ->.  The loop might be more complicated, but in any event
the sequence leads back to the original operator ->.
```

1.496 scm1570

```
Error 1570:  A linkage-specification may occur only in file scope.

Linkage-specifications are not permitted in block scopes, class scopes, or
function scopes.
```

1.497 scm1571

Error 1571: Cannot define a type in return or argument types.

A type (for example a struct tag) cannot be defined in an argument list or in the specification of the return type.

1.498 scm1572

Error 1572: object may not have the same name as its class.

A static data member, enumeration, member of an anonymous union, or a nested type cannot have the same name as its class. For more information, refer to section r.9.2 in The C++ Programming Language, Second Edition.

1.499 scm1573

Error 1573: An overloaded operator cannot have default arguments.

It is illegal to declare overloaded operators with default arguments. For example, int operator +(int=3,int=4) is not a valid declaration. For more information, refer to section r.8.2.6 in The C++ Programming Language, Second Edition.

1.500 scm1574

Error 1574: Invalid use of abstract class: class-name.

An abstract class cannot be used as an argument type, a function return type, or the type of an explicit conversion.

1.501 scm1575

Error 1575: An object of a class with a object-type may not be a member of a union.

An object of a class with constructors, destructors, or user-defined assignment operators cannot be a member of a union. For more information, refer to section r.9.5 in The C++ Programming Language, Second Edition.

1.502 scm1576

Error 1576: Error declaring 'new': reason.

operator new must have a return type of void*. Its first argument is required and must be of type size_t. For more information, refer to section r.12.5 in The C++ Programming Language, Second Edition.

1.503 scm1577

Error 1577: Error declaring 'delete': reason.

The delete function must have return type void. Its first argument must be of type void* and if there is a second argument, it must be of type size_t. No more than two arguments are permitted. For more information, refer to section r.12.5 in The C++ Programming Language, Second Edition.

1.504 scm1578

Error 1578: Initializer-clause cannot be used for class having a object-type.

A class having a constructor, a private or protected member, a base class, or a virtual function is not an aggregate and cannot be initialized by means of an initializer-clause (for example, ={10,2,10.2}). For more information, refer to section r.8.4.1 in The C Programming Language, Second Edition.

1.505 scm1579

Error 1579: Conversion to a virtual base class ('class-name') from a derived class is not allowed for member pointers.

Virtual base classes cannot be converted, explicitly or implicitly, from derived classes.

1.506 scm1580

Error 1580: Cannot return (attempted-return-type) from function returning (declared-return-type).

The return value is of a type that cannot be converted to a type required by the function's return type.

1.507 scm1581

```
Error 1581:  Function 'function-name' has an initializer.

Functions cannot be initialized, although function pointers can be
initialized.
```

1.508 scm1582

```
Error 1582:  Character array (array-name) too short for string of length
(string-length).

A character array cannot be initialized by a string that has more
characters than the array has elements.
```

1.509 scm1583

```
Error 1583:  Too many initializers for (array-name):  found n initializers.

No array can be initialized with more initializers than the array has
elements.
```

1.510 scm1584

```
Error 1584:  Too many initializers for (class-name).

No class can be initialized with more initializers than the array has
members.
```

1.511 scm1585

```
Error 1585:  Left operand of 'operator' must be type.

The left operand of the dot operator (.) must be a class object.
```

1.512 scm1586

```
Error 1586:  Type 'type' is invalid for the left operand of 'operator'.

The left operand of the arrow operator (->) must resolve to a class
pointer.
```

1.513 scm1587

Error 1587: Case label value must be a constant expression.

You have used a non-constant expression as a case label.

1.514 scm1588

Error 1588: Duplicate case label value.

The same case label occurs more than once within a switch statement.

1.515 scm1589

Error 1589: More than one default.

There is more than one default label in a single switch statement.

1.516 scm1590

Error 1590: symbol-name is not an enum.

The specified symbol was used after the enum keyword, but is not an enumeration.

1.517 scm1591

Error 1591: symbol-name is not a class, struct, or union.

The specified symbol was used after a class, struct, or union keyword, but is not a class, struct, or union.

1.518 scm1592

Warning 1592: Wide and narrow character strings concatenated, using width.

Wide characters strings are of the form L"abc", narrow characters strings are the usual "abc". These two types of strings should not be concatenated together.

For example, neither of the following statements is valid:

```
  "abc" L"def" /* wrong */
```

```
  L"abc" "def" /* wrong */
```

If the first string in the concatenation is wide, the result is wide.
Similarly, if the first string is narrow, the result is narrow.

1.519 scm1593

Warning 1593: Missing return statement.

A return statement is missing at the end of the outer block of a function,
and a return value is required.

1.520 scm1594

Warning 1594: Zero-length array used.

Zero-length arrays are allowed in classes (types class, struct, and union)
as an extension of standard C and C++. This message is only to warn you
about the use of this extension. If you do not want to see this warning,
use the ignore compiler option to turn it off.

1.521 scm1597

Warning 1597: '%s' assigned to '%s'.

A longer data type has been assigned to a shorter data type. For example,
a long has been assigned to a short. If the value assigned is a constant
which is too large to be represented by the shorter type, this is an
error. If the assigned type is a variable, this is a warning. (If the
value of an assigned constant can be represented by the shorter type,
neither a warning nor an error will be diagnosed.)

1.522 scm1610

Error 1610: Previous declaration of 'symbol' was 'attribute', this
declaration is 'attribute'.

An attribute of a symbol was declared differently in a previous
declaration. For example, the previous declaration may have been declared
with __near, but this declaration of the same symbol specified __far. Make
the declarations consistent. \em For some attributes, C++ applies a
default if the attribute was not explicitly specified in the declaration.
A declaration with a specific keyword may conflict with a previous
declaration with no keyword depending on the current defaults. The default
attributes applied by C++ depend upon the specific attribute and any
compiler options you specify.

1.523 scm1611

Error 1611: Previous declaration of 'symbol' differed in the use of 'keyword'.

An attribute of a symbol was declared differently in a previous declaration. Either the previous declaration used a keyword that is not present in this declaration or did not use a keyword that is present in this declaration. For example, the previous declaration may have been __aligned, and this declaration is not. Make the declarations consistent. \em For some attributes, C++ applies a default if the attribute was not explicitly specified in the declaration. A declaration with a specific keyword may conflict with a previous declaration with no keyword depending on the current defaults. The default attributes applied by C++ depend upon the specific attribute and any compiler options you specify.

1.524 scm1612

Error 1612: Asm function has parameter without a register.

All parameters to a function declared with __asm must be passed in an explicitly specified register.

1.525 scm1613

Error 1613: Asm function uses register '%s' more than once.

More than one parameter to an __asm function used the same register. C++ uses some registers to pass internal arguments to functions declared with __asm. If there is a conflict between the registers you specify and the registers needed by the translator, this message is displayed with an explanation of the conflict.

1.526 scm1614

Error 1614: Previous declaration of asm function used different registers, was '%s', now '%s'.

Functions declared with __asm must specify the same registers each time the function is declared.

1.527 scm1615

Error 1615: Explicit register 'register' keyword used in non-asm function.

Explicit registers can only be used on parameters for functions declared with the __asm keyword.

1.528 scm1616

```
Error 1616:  Vararg functions cannot be '__asm'.

Variable argument functions
cannot be declared with the __asm keyword.
Variable argument functions are those functions that take an
ellipsis (. . .) in their prototype.
```

1.529 slm103

```
Error 103:  Out of memory!!

slink has run out of memory.  By default, slink attempts to cache the
object modules in memory between pass 1 and pass 2.  If slink runs out of
memory, it attempts to free the cached object modules.  In some cases,
slink cannot find enough contiguous memory.  Try adding bufsize 4096 to the
slink command.  This option turns off object module caching.
```

1.530 slm425

```
Error 425:  Cannot find library library-name

The linker cannot find the specified library.  Usually, SAS/C libraries are
located in the LIB:  directory.  You may have misspelled the library name
in your slink command.
```

1.531 slm426

```
Error 426:  Cannot find object name

The linker cannot find the specified object.
```

1.532 slm443

```
Error 443:  filename is an invalid file name

The linker found an invalid character in a filename.
```

1.533 slm444

```
Error 444:  Hunk_Symbol has bad symbol-type symbol symbol-name

The object module is corrupt.  Try recompiling the object module.
```

1.534 slm445

```
Error 445:  Invalid HUNK_SYMBOL symbol-name
```

The object module is corrupt. Try recompiling the object module.

1.535 slm446

```
Error 446:  Invalid symbol type symbol-type for symbol-name
```

Either the object module is corrupt, or slink has found a symbol type that it does not recognize. Try recompiling the object module. For a list of valid symbol types, refer to the description of object file structure in The AmigaDOS Manual, 3rd Edition.

1.536 slm447

```
Error 447:  filename is a load file
```

The file specified is an executable module instead of and object module.

1.537 slm448

```
Error 448:  filename is not a valid object file
```

The file specified is not an object module. The file may be a C source file or a corrupt object module.

1.538 slm449

```
Error 449:  No hunk_end seen for filename
```

The object module is corrupt. Try recompiling the object module.

1.539 slm450

```
Error 450:  Object file filename is an extended library
```

A library file was specified as an object module. Add the library keyword in front of the library name.

1.540 slm501

```
Error 501:  Invalid Reloc 8 or 16 reference
```

An 8- or 16-bit relocation address cannot reach its destination. The SAS/C Compiler does not generate 8-bit relocation records, but third party products may use them. The object module is probably corrupt. Try recompiling your source file. If recompiling the file does not correct the error, contact the Technical Support Division.

1.541 slm502

```
Error 502:  function-name symbol - Distance for Reloc16 greater than 32768
```

The distance from the point where the function is called to the function itself is greater than 32767 bytes. The function cannot be referenced with a 16-bit address field. Normally, slink tries to insert an ALV (Automatic Link Vector) at the end of the calling module, but if the module has more than 32K of code, the relocation may not reach the ALV. An ALV is the instruction used to reach functions that would otherwise be too far away. Compile the file with code=far, or declare the function with the __far keyword.

1.542 slm503

```
Error 503:  function-name symbol - Distance for Reloc8 greater than 128
```

The distance from the point where the function is called to the function itself is more than 128 and so the function cannot be referenced with an 8-bit address field. The SAS/C Compiler does not generate 8-bit relocation records, but third party products may use them.

1.543 slm504

```
Error 504:  variable-name symbol - Distance for Data Reloc 16 greater than
32768
```

The distance for the 16-bit relocation is too far. You may see this message if a data item (such as a structure or an array) is larger than 64K or if the total amount of data in your program is greater than 64K. To correct the problem, you can either break the data item down and make it smaller than 64K, or you can place the __far keyword on the definitions of one or more data items to force all references to those items to be 32-bits. Using the __far keyword reduces the amount of data in the near data section to less than 64K. You can also compile your program with the data=far option. However, using data=far increases code size and execution time.

1.544 slm505

```
Error 505:  variable-name symbol - Distance for Data Reloc8 greater than
128
```

The distance for the 8-bit relocation is too far. The SAS/C Compiler does not generate 8-bit relocation, but third party products may use them.

1.545 slm506

```
Error 506:  Can't locate resolved symbol symbol-name
```

If this message appears, please contact the Technical Support Division. See Chapter 3, "Getting Help," for a complete list of items that you need to provide to the Technical Support staff.

1.546 slm507

```
Error 507:  Unknown Symbol type symbol-type, for symbol symbol-name
```

Either the object module is corrupt, or slink has found a symbol type that it does not recognize. Try recompiling your source file. For a list of valid symbol types, see the description of object file structure in The AmigaDOS Manual, 3rd Edition.

1.547 slm508

```
Error 508:  Symbol type symbol-type unimplemented
```

Either the object module is corrupt, or slink has found a symbol type that it does not recognize. Try recompiling your source file. For a list of valid symbol types, see the description of object file structure in The AmigaDOS Manual, 3rd Edition.

1.548 slm509

```
Error 509:  Unknown hunk type symbol-type in Pass2
```

If this message appears, please contact the Technical Support Division. See Chapter 3, "Getting Help," for a complete list of items that you need to provide to the Technical Support staff.

1.549 slm510

```
Error 510:  symbol-name symbol - Near reference to a data item not in near
data section
```

The linker expected the specified symbol to be in the near data section, but the symbol is located in either the far data section, chip memory, or the code section.

You may have defined a variable as __far with the __far keyword in one module and externally declared the same variable in another module without the __far keyword. If so, the module with the external declaration attempted to reference the data as near. To correct the problem, the definition and all declarations of the data item must be specified with the same keywords.

If you compiled your file with the data=auto option, the compiler can place variables into the far data section if necessary. The compiler may have placed a variable in one module in the far section and a variable in another module in the near section. To correct this situation, either stop using the data=auto option, or use the __far keyword (if necessary) on the definition and all declarations of the variable that caused the error.

1.550 slm512

```
Error 512:  Invalid branch to function-name in overlay node module-name
```

The linker detected a branch in an overlay node that calls another overlay node at the same level. This type of branch is not supported by the overlay manager.

1.551 slm513

```
Error 513:  Multiple NTRYHUNK segments not permitted
```

The NTRYHUNK is the root overlay node. No other hunks should have this name.

1.552 slm514

```
Error 514:  Overlay manager _ovlyMgr is undefined
```

This _ovlyMgr function is located in SAS/C libraries. To use the SAS/C overlay manager, link with sc.lib or scs.lib. You can also write your own overlay manager. See Chapter 8, "Compiling and Linking Your Program," for information on creating your own overlay manager.

1.553 slm515

```
Error 515:  An ALV was generated pointing to data variable-name symbol
```

A 16-bit relocation could not reach its destination, so slink inserted an ALV (Automatic Link Vector) instruction, and then determined that the destination is not in a code hunk. To correct the problem, make the reference a 32-bit reference. This error may be generated for object code produced by third-party assemblers and compilers

1.554 slm516

```
Error 516:  Attempt to merge BSS with CODE or CODE/DATA
```

Either you have attempted to merge the far BSS section with the far data section, or you have attempted to merge the near or far BSS section with the code section. You can merge a BSS section with the __MERGED section only. You will see this message if you have assigned the same name to either the far BSS section and the far data section or the far BSS section and the code section.

If you get this error in another situation, please call the Technical Support Division.

NOTE: Do not name the code section __MERGED.

1.555 slm600

```
Error 600:  Invalid option command
```

The option listed is not a valid slink option.

1.556 slm601

```
Error 601:  option option specified more than once
```

The option has been specified twice.

1.557 slm602

```
Error 602:  Unable to open output file filename
```

slink cannot open the output file for write access. Another program such as CodeProbe may have left a lock on the file. Alternatively, the protection bits may prohibit write access to the file, or the name specified may be a directory.

1.558 slm603

```
Error 603:  string is not a valid number
```

The linker found a non-numerical character in a field where it expected to find a number.

1.559 slm604

```
Error 604:  with file is not readable
```

The with file may contain binary characters or may be locked.

1.560 slm605

```
Error 605:  Cannot open with file filename
```

The with file does not exist or may be locked.

1.561 slm607

```
Error 607:  No FROM/ROOT files specified
```

You did not specify an object module, or if you are using overlays, the root node did not contain any objects.

1.562 slm608

```
Error 608:  Premature EOF encountered
```

The object module is corrupt. Try recompiling the source file.

1.563 slm609

```
Error 609:  Error seeking in file filename
```

The object module is corrupt. Try recompiling the source file.

1.564 slm610

```
Error 610:  module-name has no parent in overlay tree
```

The with file specifies an overlay with no parent node.

1.565 slm611

```
Error 611:  Reloc found with odd address for symbol symbol-name, file
filename
```

The linker has found a relocation record with an odd address. This message can only be generated if your program is written in assembler and is usually caused by placing an odd length character string in the code section.

1.566 slm612

```
Error 612:  MERGED Data relocation to non-code section in Overlay node
            Reference at offset hex-address in module-name, To Unit
            module-name
```

The linker found a relocation from the near data section, which is placed in the root node, to the data section of an overlay node. This type of relocation is illegal.

1.567 slm613

```
Error 613:  MERGED Data relocation to static function is not resolvable by
            Overlay Manager.
            Reference at offset hex-address in filename, To Unit filename
```

The overlay manager cannot resolve an indirect function call from the root node to a static function in a overlay node. To correct the problem, remove the static keyword from the function declaration.

1.568 slm614

```
Error 614:  More than one MERGED data section found
```

Only one near data section is allowed. This error occurs only when using slink to strip debug information from an executable generated by a third party linker. If you get this message, do not use slink to strip debug information from this executable.

1.569 slm615

```
Error 615:  Code hunk named __MERGED
```

slink merges all hunks with the name __MERGED and performs relocations to this hunk relative to register A4. Do not name the code hunk __MERGED .

1.570 slm616

```
Error 616:  ALVs were generated
```

You have linked with the noalvs linker option, but the linker could not resolve all relocations without generating ALV instructions. Either the total code size is greater than 32K, or there are multiple code hunks. The executable will run, but the code section is not totally PC-relative.

1.571 slm617

```
Error 617:  MERGED data section greater than 64K
```

If your program does not generate any additional messages, then the program will still run. However, a 16-bit data relocation record may not be able to reach its target, and if this happens, slink will generate an error.

1.572 slm618

```
Error 618:  Multiple OVERLAY usage--previous occurrences were ignored
```

You can only specify one overlay manager.

1.573 slm619

```
Error 619:  Ignoring null OVERLAY list
```

You did not specify any files in the overlay list.

1.574 slm620

```
Error 620:  Missing '#' at end of OVERLAY list
```

Enter a pound sign (#) at the end of the overlay tree. For more information on creating overlay trees, see Chapter 8, "Compiling and Linking Your Program."

1.575 slm621

```
Error 621:  Conflicting integer sizes found
```

Some modules were compiled using short integers and some were compiled using long integers. You cannot mix integer sizes in an executable. Alternatively, you may have linked in the wrong version of the library.

1.576 slm622

```
Error 622:  Conflicting math types found
```

You may have compiled the various object modules with conflicting math options. All object modules must be compiled with the same math library. Alternatively, you may have linked in the wrong math library.

1.577 slm623

```
Error 623:  Regargs function function called through overlay manager.
Parameters passed in registers to this function will be destroyed.  Use
NEWOCV option.
```

The regargs function passes parameters in scratch registers, but the overlay manager does not preserve scratch registers. You cannot call a regargs function across an overlay node, unless you use the newer alternative overlay manager, which is invoked with the NEWOCV option to slink.

1.578 slm624

```
Warning 624:  Absolute reference to symbol module:  file filename
```

If you reference far data in a module linked with cres.o or when you are generating a shared library, your program may function improperly unless the data are read-only. slink cannot determine if the reference is read-only, so it generates this warning.

1.579 slm625

```
Error 625:  Proper math library has not been included
```

You have not linked in the math library that is needed for the current math options.

If you compile with the link option, you need to include the appropriate math option to specify the math library with which you want to link. If you are using the slink command to link your file, you need to link with a math library as well as sc.lib.

1.580 slm626

```
Error 626:  Libcode used on module module
```

You have compiled the module with the libcode compiler option, but the module is being linked into an executable. Use the libcode option for generating shared libraries only.

1.581 slm627

```
Error 627:  Near references found in executable that has a module compiled
with FARONLY option

You cannot mix modules compiled with data=faronly and modules that refer to
near data.
```

1.582 HELP

```
You have reached this Help window by either clicking on the Help
button or by hitting the Help key within the SAS/C Help utility.
Unlike other help topics present in the SAS/C Help utility, the Help
help topic opens its own window.  You must close this window by
clicking on the close gadget or hitting escape before returning to
the SAS/C help utility.  You cannot hit the Retrace button to return.

To quit the SAS/C Help utility, select Quit from the Project menu or
click on the close gadget.  You may also hit escape.

Most help screens will display one or more buttons as part of the text.
Clicking on these buttons will provide further information on the topic
listed on the button.  You can also reach these help topics through the
main Contents screen or one of its sub-screens.

In addition, double-clicking in the help window will bring up a help
screen for the word under the mouse cursor, if such a help screen
exists.

While in the SAS/C Help utility, you may retrace your steps through the
help screens you have selected by clicking on the Retrace button.

The Browse buttons will move you forward and backwards between help
screens.  The help screens are usually arranged alphabetically by
command or topic.
```