Keil C51 Bugfix Report

Modifications, Enhancements and Corrections in Keil C51 Compiler for Version 6.xx

Contents:

Modifications and Corrections in C51 Compiler from Version 6.00 Differences to Version 6.02

Part 1: C51 Version 6.02

☑ Corrected: ASMPARMS causes wrong parameter passing

```
#pragma ASMPARMS
extern unsigned char uc_BF_SecCmpLong(unsigned long, unsigned long);
#pragma NOASMPARMS

void v_Test(void) {
   unsigned long idata ulCheckSumI=0x11223344;
   unsigned long xdata ulCheckSumX=0x55667788;
   unsigned long code  ulCheckSumC=0xAABBCCDD;

   uc_BF_SecCmpLong(ulCheckSumI,ulCheckSumX);   // wrong
   uc_BF_SecCmpLong(ulCheckSumI,ulCheckSumC);   // wrong
   uc_BF_SecCmpLong(ulCheckSumX,ulCheckSumI);   // o.k.
   uc_BF_SecCmpLong(ulCheckSumX,ulCheckSumC);   // o.k.
   uc_BF_SecCmpLong(ulCheckSumC,ulCheckSumI);   // o.k.
   uc_BF_SecCmpLong(ulCheckSumC,ulCheckSumX);   // o.k.
   return;
}
```

☑ Corrected: Unused results from crol and cror causes internal error

```
extern unsigned char _crol_    (unsigned char, unsigned char);

void test( void ) {
  unsigned char h;
  unsigned char i;

  i = 0x01;
  for( h = 0x80; i != 0x01; i--) _crol_( h, 1 );
}
```

☑ Corrected: Access via Pointer cast fails

```
extern unsigned char pdata ucaBuffer [0x100];

void test (void)  {
  *((unsigned int pdata*)&ucaBuffer)= 0;  // OK
  *((unsigned int *)&ucaBuffer)= 0;       // OK
  *((unsigned int xdata*)&ucaBuffer)= 0;  // fails
}
```

☑ Corrected: mSpace via typedef fails on arrays

```
typedef      unsigned char UInt8;
typedef const UInt8 code MyArray;

MyArray  V1[ ] = {1,2,3,4,5,6,7,8,9};     // Error: placed in data space
MyArray  V2[5];
MyArray  V3 = 0x10;
MyArray  V4;

typedef UInt8   xdata xUC;
typedef UInt8   idata iUC;
```

```
typedef UInt8    data dUC;

xUC  x1,x2,x3;
iUC  i1,i2,i3;
dUC  d1,d2,d3;

void main (void) {
 ;
}
```

☑ Corrected: Scope-Info may be incorrect with OT(8/9)

```
Debug Information confuses Emulators. Happens when a function with parameter(s) gets
one ore more preheaders, most likely with ot(8) and ot(9)
```

☑ Corrected: Following programs generates a MOV #const,#const instruction

```
typedef unsigned char U8;
typedef unsigned short  U16;

typedef struct {
 U8 a :4;
 U8 b :4;
 U16  c;
}STRUCT_A;

STRUCT_A var;

void main (void)  {
  var.b += 1;
  while (1);
}
```

☑ Corrected: Incorrect code with OT(9) because of Rb(n)/Using

```
#define STR(x) #x
#define XSTR(x) STR(x)

#define IDEF    Bull.h

#include XSTR (IDEF)

typedef struct  {
   unsigned char nxt;
   unsigned int  wt;
}T;

#define SIZE 10

unsigned int pdata g1;
T xdata Arr[SIZE+1];
#pragma rb(0)
void f1(unsigned char v1,unsigned int v2)  {
   unsigned char bv1;
   unsigned char bv2;

   Arr[v1].wt = v2 + g1;

   bv1 = Arr[SIZE].nxt;
   bv2 = SIZE;
   Arr[SIZE].wt = g1;
```

```
   if (Arr[v1].wt < g1)
   {
      while ((bv1 != 0xFF) && (g1 < Arr[bv1].wt))
      {
         bv2 = bv1;
         bv1 = Arr[bv1].nxt;
      }
   }

   while ((bv1 != 0xFF) && (v2 > (Arr[bv1].wt-g1)))
   {
      bv2 = bv1;
      bv1 = Arr[bv1].nxt;
   }

   Arr[v1].nxt = bv1;
   Arr[bv2].nxt = v1;
}

#pragma rb(2)
void f2(void) interrupt 4 using 2  {
}
```

☑ Corrected: General Protection Foults on Syntax Errors

☑ Corrected: SRC File is incorrect and cannot be assembled

```
sfr P1 = 0x90;
sbit page = P1^0;   // page is a reserved word in A51

void main (void)  {
  page = 1;
}
```

☑ Corrected: Only with OT(9) & ACALLOPT: Acall-target has wrong offset in object file

```
SampleFile: Rs232.i
```

☑ Corrected: Register allocation may fails in nested function calls

```
Test-File: C51TEST\V6BUGS\B1.C #695
```

☑ Corrected: address calculation on structs wrong

```
typedef struct  {
  unsigned char a;
  unsigned char b;
}TEST_STRUCT_1;

typedef struct  {
  TEST_STRUCT_1 c;
  unsigned char d;
} TEST_STRUCT_2;

unsigned char index;

TEST_STRUCT_2 xdata array[10];
```

```
void main (void) {
  unsigned char *char_ptr;
  TEST_STRUCT_1 *struct_ptr;
  char_ptr   = &(array[index].d);  // address calculation OK
  struct_ptr = &(array[index].c);  // address calculation wrong
}
```

☑ Corrected: MOVB instruction used also on register based variables

```
Note: only relevant if MODE2 is used.
```

☑ Corrected: With Mode2 Directive only: 'Mov dir,@dptr' instruction not recognized

---

☑ Corrected: Following code generates a MOV R6,DPTR instruction

```
unsigned char i;

void main (void)  {
  unsigned char code *ep;

  ep = 0;
  while (ep < 0x4000)  {
    if (*ep != 0xFF) i+= *ep;
  }
}
```

☑ Corrected: Extra Line number causes confusion while debugging

```
unsigned int dummy;
void testfunc(void) {
}
void main(void) {
  while (1)  {
    dummy++;
    testfunc();
  }
}   // Problem: for this line a LineNo is created, no RET follows.
```

☑ Corrected: Common Tail Merging with OT(8) should not cover overlayable calls

```
#pragma CD SB NOIP ROM(Compact) DB OE SM
// functions that have overlayable segments should not be merged
// into another function call, since data overlaying gets inefficient.

unsigned char called1(void)  {
  unsigned char tab[10];
  unsigned char i;
  unsigned char result;
  for(i=0;i<10;i++) result=result+tab[i];
  return(result);
}

unsigned char called2 (unsigned char pParam)  {
  unsigned char tab[10];
  unsigned char i;
  unsigned char result;

  result=pParam;
```

```
  for(i=0;i<10;i++)  result=result+tab[i];
  return(result);
}

void main(void)  {
  unsigned char tempByte;
  tempByte=3;
  called1();
  called2(tempByte);
  tempByte=4;
  called1();
  called2(tempByte);
  tempByte=5;
  called1();
  called2(tempByte);
  tempByte=6;
  called1();
  called2(tempByte);
}
```

☑ Corrected: Register Parameter Passing Fails

```
#define FALSE 0
#define TRUE (!FALSE)
#define NULL 0
typedef unsigned char       uBYTE;
typedef bit BOOL ;
extern uBYTE xdata *xgzgr1;

typedef struct {
  uBYTE l;
  uBYTE d[40];
} CID;

typedef struct {
  uBYTE a;
  CID cid;
} tSE;

BOOL asm_Abl( uBYTE xdata *KGK_zgr, uBYTE xdata *CID_zgr);
tSE xdata SE;

CID xdata* GetCID(void) {
  if(SE.cid.l==0)return(NULL);
  return(&SE.cid);
}

BOOL KA_Abl(uBYTE xdata *KGK_zgr)   {
 xgzgr1 = (uBYTE xdata *) GetCID();
 if (!xgzgr1)   return FALSE;

// im SRC File erkennt man die falsche Registerbelegung,
// R4/R5 ist mit R6/R7
// bei der Parameteruebergabe zu asm_Abl() getauscht!
 return  asm_Abl(KGK_zgr, xgzgr1);
}
```

☑ Corrected: Order of Symbols differs in SRC and OBJ Mode

```
// Optimizer uses the OBJ Mode Offsets, therefore needs to be identical in SRC Mode
/* Offset in OBJ listed in Symbol table, Offset in SRC Mode is order of Sym
padvalue_sta . . . . . .  PUBLIC   IDATA  U_CHAR   0001H  1
pinlevel_invers_sta. . . . . .  STATIC   IDATA  ARRAY    0002H  4
pinlevel_sta . . . . . .  STATIC   IDATA  ARRAY    0006H  4
```

```
tasten_sta . . . . . .  STATIC   IDATA  U_CHAR   000AH  1
tastenabfrage. . . . . .  STATIC   CODE   PROC     0000H  -----
  tastenzustand_sta. . . . . .  STATIC   IDATA  U_CHAR   0000H  1
eventvalue_sta . . . . . .  PUBLIC   IDATA  U_CHAR   000BH  1
*/
unsigned char idata padvalue_sta;
unsigned char idata eventvalue_sta;
static unsigned char idata tasten_sta;
static unsigned char idata pinlevel_sta[4];
static unsigned char idata pinlevel_invers_sta[4];

static void tastenabfrage(void) interrupt 1   {
  static unsigned char idata tastenzustand_sta;
}
```

☑ Corrected: First line number does not match on interrupt f()'s

```
#pragma LARGE OE DB SB
xdata char zw;
xdata char tim1;

sfr SP = 0x80;

int i1test (int i1, int i2)
{
  return ((i1 * i2) + (i1 / i2) + SP);
}

void timer1(void) interrupt 3 using 2   // no line here becaus of push'es
{

  i1test (1, 2);
  i1test (1, 2);
  i1test (1, 2);
  zw = SP;
  tim1++;
}

void test1 (void)
{
  ++tim1;
  zw = 0x22;
}
```

☑ Corrected: Following Code accesses wrong idata variables

```
unsigned int data value;
unsigned int idata itv;

void main (void)  {
  if (value != itv)  {
    itv = value;
  }
}
```

☑ Corrected: Function argument not promoted in ternary operator

```
#pragma ROM(COMPACT) SMALL DEBUG OBJECTEXTEND CD SB DB OE OT(6,Size)

extern void check1(char *cptr) ;
extern void check2(char abyte) ;
extern void dothis(void) ;
```

```
void test(char xdata *cptr) {
  char isok = 1;

  if(isok) dothis();
  check1 (cptr);
  isok ? check1(cptr) : check2(isok);  // 'cptr' is not cast to generic ptr
}
```

☑ Corrected: Missing syntax error / variable not defined

```
char _cVariable;   // variable defined with '_'

void func (void)  {
  cVariable = 0;   // does not generate syntax error
}
```

☑ Corrected: Potential Data Overlaying Problem

```
Test File: \C51TEST\V6BUGS\A26.C
```

☑ Corrected: Unbalanced PUSH with volatile load/store

```
#pragma LARGE
extern unsigned char SYS_ChkEvent(unsigned int sys_out);
unsigned int sys_out;
extern void dispatch_event(void);
volatile unsigned char DESSTAT;

void main(void) {
  DESSTAT = DESSTAT;       // generates unbalanced PUSH
                           // because STORE is removed
  if (SYS_ChkEvent(sys_out) != 0) dispatch_event();
}
```

☑ Enhancement: Switch/case code improved

```
Test File: \C51TEST\V6BUGS\A27.C
```

☑ Corrected: Pointer access to wrong variable

```
void exor_bytes (char *a, char *io )  {
  unsigned char i;

  for ( i = 0; i < 8; i++ )
    io[i] ^= a[i];   // generates a[i] ^= io[i];
}
```

☑ Corrected: Incorrect common subexpr. with MODE2 and intrinsic-copy

```
#pragma MODE2 code

void _copy_s2i_ (void idata *dest, unsigned char src, unsigned char len);

sfr SHIELD1 = 0xDA;
sfr SHIELD2 = 0xDB;

unsigned char function_copy_bug(void)  {
  idata unsigned char buffer[2];
  xdata unsigned char ziel[4];
```

```
  _copy_s2i_(buffer, SHIELD1, 2);
  ziel[0] = buffer[0];
  ziel[1] = buffer[1];

  _copy_s2i_(buffer, SHIELD2, 2);
  ziel[2] = buffer[0];
  ziel[3] = buffer[1];
  return (1);
}
```

☑ Corrected: Compiler does not report double definition on bit/sbits

```
#define extern
extern   bit    Auto;          // Define some bits
extern   bit  Manual;
extern   bit Reverse;

bdata unsigned char Group;   // Force bits into bdata area for Tx of byte
sbit Auto = Group^1;         // Redefine bits - should get a compiler error
sbit Manual = Group^2;
sbit Reverse = Group^3;

void main (void)  {
  Group   = 1;         // Auto should get a value of 1 now, but does not??
  Auto    = 1;         // Now Auto is equal to 1
  Reverse = Auto;
  Manual  = 0;
}
```

☑ Corrected: FATAL ERROR: GLOBAL OPTIMIZATION on following code

```
#pragma LARGE CODE

main ()  {
  unsigned char *p_klok;
  unsigned char maxfart;
  signed int ny_vh, ny_fb;

  if ((ny_vh<=-60)||(ny_vh>=60))  {
    ny_fb=(ny_fb*5)/10;
    ny_vh=(ny_vh*5)/10;
    if (ny_fb>110)  ny_fb=110;
  }
  p_klok[8] = maxfart;
}
```

☑ Corrected: FATAL ERROR: GLOBAL OPTIMIZATION on following code

```
#pragma large MODE2
extern void f(char code*, char code*, int);
extern char buf[];

char g(void) {
  char    a;

  f((char code*)buf + a, (char code*)buf, *(&(&buf[5])[a]));
  return(1);
}
```

☑ Corrected: Optimize(9) May Fail

☑ Corrected: 'Undefined identifier' error message

```
/*
 * C2N090: *** ERROR 202 IN LINE 29 OF E388.C: undefined identifier
 */

extern void exit(int);

int f1(int i) reentrant  {
      return i;
}
int f2(int f) reentrant  {
      return f;
}
void main()  {
      int (*ap[2])(int) reentrant;
      int i,j;

      ap[0] = f1;
      ap[1] = f2;
      i = (*ap[0])(1);
      j = (*ap[1])(2);
      if ( i != 1 ) exit(1);
      if ( j != 2 ) exit(2);
      exit (0);
}
```

☑ Corrected: FIXUP Error with OPTIMIZE(9) ROM(COMPACT) & ACALLOPT

```
An AJMP to a common block gets the SI of the current code segment, should get the SI of
the common block.
Testfile: C51test\v6bugs\e.c
```

☑ Corrected: potential destroy of parameters for indirect called functions with more than one parameter

```
#pragma code
#pragma small
#pragma optimize(6,size)

typedef void (code * TPTR)(unsigned char xdata *, void xdata *); //Pointer to task

typedef struct  {
  unsigned char xdata *status;   //  10
  TPTR                 tcode;   // Task code
  void          xdata  *tdata;   // Task data
} SCELL;

xdata SCELL   tsk[10];                // Array of task description cells
extern unsigned char current_task;

void scheduler (void)  {
  (*(tsk[current_task].tcode))(tsk[current_task].status,
                               tsk[current_task].tdata);
}
```

☑ Corrected: General Protection Fault on Syntax Errors

```
typedef int TRecorder[8][256];
#define iaRecorder (*((TRecorder xdata *)0xEC00))

void Recorder (void)  {
  static unsigned char bRecIndex = 0;

  int xdata * pRecorder;
  pRecorder = &(iaRecorder[0][bRecIndex++]);
}
```

☑ Corrected: Func Address cannot be in common code, linker generates a potential Warning 13

```
int xdata i;
long xdata l;
long data ld;
void (*fp) (void);

void func (void)  {
  char arr[10];
  l = ld;
}

void func1 () {
  i = 0;
  fp = func;
  l = ld;
}

void func2 () {
  i = 0;
  fp = func;
  l = ld;
}

void main (void)  {
  func ();
  func1 ();
  func2 ();
}
```

☑ Corrected: Addresse instead of pointer content used

```
unsigned char code * idata iip2;
unsigned char code * idata iip;
unsigned char code * dip;

void xmain (void)  {
  iip2 = dip;
  iip = iip2+ 1;   // bad
}
```

☑ Corrected: USING attribute not transferred to common function

☑ Corrected: Only in MODE2 - SLE66 Mode, unbalance PUSH/POP generated

```
#pragma mode2
static unsigned char bNMI;
```

```
static void NMI() interrupt 0  {
      bNMI = 1;
}
static int nTestCase;

static void CheckPeripheralProtection(unsigned char bProtOn) {
  bProtOn = 1;
}

static void TestPeripheralProtection()  {
  CheckPeripheralProtection(0);
}

static void SystemMode() interrupt 1  {
  switch(nTestCase)  {
      case 0: CheckPeripheralProtection(0);
                return;
      case 1:     CheckPeripheralProtection(0);
                return;
  }
}
```

☑ Corrected: Problem with integer shift right operation >> 8

```
char ProcTest() {
  char data a_c  = 0xAA;       //init (unimportant)
  unsigned int data b_ui = 0x1234;
  a_c = (char)(b_ui>>8);       //store high-order char wrong register used
  return (a_c);
}

// 2. example
unsigned int dummy;
void main (void) {
  unsigned int ui_hilfsvar;

  ui_hilfsvar = dummy;
  ui_hilfsvar += TL2 + 28;
  TL2 = (unsigned char)ui_hilfsvar;
  TH2 = (unsigned char)( ui_hilfsvar >> 8 ); // hier wird der falsche Wert geschrieben
                                             // R6 ist falsch. richtig waere A
  while(1);
}
```

☑ Corrected: P (=parity bit) is not a reserved word, it is an sfr bit

```
#include <Dallas/REG320.h>

sbit TB8_1 = SCON1^3;

void main (void)
{
   #pragma ASM
      mov C, P      // this does not translate, because it was a reserved word
      mov TB8_1, C
   #pragma ENDASM
}
```

☑ Corrected: Data Segment Offset in SRC Mode is incorrect

```
#pragma db sb cd src
#define uBYTE unsigned char
```

```
uBYTE func1(uBYTE p1)  {
  uBYTE i0, i1, i2, i3, i4, i5, i6, i7;
  i0 = i1 = i2 = i3 = i4 = i5 = i6 = i7 = p1;
  {
    uBYTE m1 = 0xff;
    {
      uBYTE n1 = 0xff;
    }
  }
  return i0 | i1 | i2 | i3 | i4 | i5 | i6 | i7;
}

uBYTE func0(void)  {
  uBYTE i0, i1, i2, i3, i4, i5, i6, i7;
  i0 = i1 = i2 = i3 = i4 = i5 = i6 = i7 = 0x00;
  {
    uBYTE m1;
    m1 = 0xff;
    {
      uBYTE n1 = 0xff;
    }
  }
  return i0 | i1 | i2 | i3 | i4 | i5 | i6 | i7;
}

void main(void)  {
  uBYTE idx1;
  idx1 = func1(0);
}
```

☑ Enhancement: Warning 284 add; 'symbol': in overlayable space, function no longer reentrant

```
int test (void)  reentrant  {
  int xdata i;
  return (i);
}
// *** WARNING C284 IN LINE 4 OF X.C: 'i': in overlayable space, function no longer
reentrant

int test2 (void)  reentrant  {
  static int xdata i;
  return (i);
}
```

☑ Corrected: Address instead of variable content used

```
extern void copy_code_to_pdata (void pdata *, void code *, unsigned char);

unsigned char idata i;
unsigned char code* ucpBuffer;
unsigned int idata j;
unsigned char xdata* ucaBuffer;

extern void func (void);

void main (unsigned int uioffs)  {
  func ();
  j = uioffs;
  copy_code_to_pdata(ucaBuffer,(ucpBuffer+j+1),i);
}
```

☑ Corrected: OPTIMIZE (8) and OPTIMIZE(9) Using for ARx access not transferred

```
#pragma small code debug oe optimize(9,speed)
#include <string.h>

extern code unsigned char BitTab[];
unsigned char gl1;
unsigned char gl2;
unsigned char xdata gl3[10];
unsigned char xdata glx2;
unsigned char xdata gl4;

#pragma rb(3)
unsigned char f2(unsigned char *p1)
{
   unsigned char i,j,k;

   unsigned char plptr;
   unsigned char *paptr;

   if(gl2)
   {
      paptr = gl3;
      j=gl1>>3;
      k=p1[j];

      if (glx2)
      {
        if(!((memcmp(p1,paptr,j)) || ((k^*(paptr+(j))) & (BitTab[glx2&0x7]))))
        {
           gl1=0x8;
           return 0xFF;
        }
        else
           return 0;
      }
      else
      {
        if(memcmp(p1,paptr,5))
        {
           gl1=0x8;
           return 0xFF;
        }
        else
           return 0;
      }
   }

   paptr = &glx2;

   for(i=0; i<10; i++)
   {

     plptr = gl3[i];
     if(((gl4==0)||(gl4==(i+1))))
     {
        if(plptr!=0xFF)
        {
          j=plptr>>3;
          k=p1[j];
           if(!((memcmp(p1,paptr,j)) || ((k^*(paptr+(j))) & (BitTab[plptr&0x7]))))
           {
              gl1=plptr;
              return i+1;
           }
```

```
        }
      }
      paptr += 5;
    }
    return 0;
}

#pragma rb(0)
unsigned char f2_b0(unsigned char *p1)
{
   unsigned char i,j,k;

   unsigned char plptr;
   unsigned char *paptr;

   if(gl2)
   {
      paptr = gl3;
      j=gl1>>3;
      k=p1[j];

      if (glx2)
      {
        if(!((memcmp(p1,paptr,j)) || ((k^*(paptr+(j))) & (BitTab[glx2&0x7]))))
        {
           gl1=0x8;
           return 0xFF;
        }
        else
           return 0;
      }
      else
      {
        if(memcmp(p1,paptr,5))
        {
           gl1=0x8;
           return 0xFF;
        }
        else
           return 0;
      }
   }

   paptr = &glx2;

   for(i=0; i<10; i++)
   {

     plptr = gl3[i];
     if(((gl4==0)||(gl4==(i+1))))
     {
        if(plptr!=0xFF)
        {
          j=plptr>>3;
          k=p1[j];
           if(!((memcmp(p1,paptr,j)) || ((k^*(paptr+(j))) & (BitTab[plptr&0x7]))))
           {
              gl1=plptr;
              return i+1;
           }
        }
     }
     paptr += 5;
   }
   return 0;
}
```

☑ Corrected: Incorrect Code generated when NOAREGS directive active

```
#pragma NOAREGS

sbit STAT_LED     = 0x97;
unsigned char idata led_flash_cnt;
unsigned char idata led_flash_mode;

unsigned int code flash_pattern[] = {0xcccc,0xff00,0xaaaa,0xaaff};

void led_flashing(void){
  if (6) {
    SAT_LED = ( flash_pattern[led_flash_mode] & (0x01 << led_flash_cnt));
    led_flash_cnt = ( ++led_flash_cnt & 0x0f );
  }
}
```

☑ Enhancement: Code Optimization: Two ANL instructions are generated

```
typedef unsigned char U8;
typedef unsigned short  U16;

typedef struct {
 U8 a :4;
 U8 b :4;
 U16  c;
}STRUCT_A;

STRUCT_A var;

void main (void)  {
  var.b += 1;
  while (1);
}
```

☑ Enhancement: Code Optimization: Long inc and add+1 different; add+1 is more efficient

```
unsigned long TmpDWord1;
void main (void)  {
  TmpDWord1 = TmpDWord1 + 1;
  TmpDWord1++;
}
```

☑ Corrected: Incorrect code on following example

```
#pragma large

unsigned int i;
unsigned char xdata *xp;

void called1(unsigned char nSize)  {
  unsigned char h_adr;
  *((unsigned char xdata *)((h_adr<<2)+nSize-1)) = 0; // Doesn't work on chip
}
```

☑ Corrected: Missing Syntax Error on following code

```
unsigned int x;

void main (void)  {
  (unsigned char) (x) = 2;         // Ok.
```

```
  (unsigned char) (x & 0xff) = 2;  // missing error not a lvalue
}
```

C51 V6.02 29. May 2000 Release Completed

Modifications and Corrections in C51 Compiler from Version 6.02 Differences to Version 6.03

Part 2: C51 Version 6.03

☑ Corrected: OT(9) Problem with 'using'

```
sfr P0=0x80;
sfr P4=0x90;

#define ClrBit(v,b)  (v &= ~(1 << b))
#define SetBit(v,b)  (v |=  (1 << b))

unsigned char can, flag;

static func1 (void)  using 1  {
  if (flag)  ClrBit(P0,can);
  else       SetBit(P0,can);
  if (flag)  ClrBit(P4,can);
}
```

☑ Enhancement: Code Optimization for bit complement

```
#include <reg52.h>

unsigned char bdata HallPattern;
sbit  Hall_0 = HallPattern^0;
sbit  Hall_1 = HallPattern^1;
sbit  Hall_2 = HallPattern^2;

unsigned char x1,x2;

unsigned char NextHallPattern;

void INT_viIsrEx0(void) interrupt 0 using 1   {
    x1 = 0;
    x2 = 0;

    Hall_0 ^= 1;// P3_2 transition detection

    if (HallPattern != NextHallPattern) {
        TL2 = 0xfa;
        TH2 = 0xfe;
    }
}
```

☑ Enhancement: Code Optimization for bit complement

```
#include <reg52.h>

unsigned char bdata HallPattern;
sbit  Hall_0 = HallPattern^0;
```

```
sbit  Hall_1 = HallPattern^1;
sbit  Hall_2 = HallPattern^2;

unsigned char x1,x2;

unsigned char NextHallPattern;

void INT_viIsrEx0(void) interrupt 0 using 1   {
    x1 = 0;
    x2 = 0;

    Hall_0 ^= 1;// P3_2 transition detection

    if (HallPattern != NextHallPattern) {
        TL2 = 0xfa;
        TH2 = 0xfe;
    }
}
```

☑ Enhancement: Code Improvement on following sample

```
#pragma LARGE
#include <stdio.h>

extern char code * code ptext[100][5];

unsigned char LANGUAGE;
char xdata Druck[100];
unsigned char i;
unsigned char MerkEE1;


void main (void)  {
  sprintf( (char*) Druck[i++], "%-17.17s:%22.22s\n",
  ptext[60][LANGUAGE],
  (MerkEE1 & 0x04) ? ptext[62][LANGUAGE] : ptext[61][LANGUAGE],
  (MerkEE1 & 0x02) ? ptext[61][LANGUAGE] : ptext[63][LANGUAGE]);
}
```

☑ Corrected: Wrong register used in long shift argument

```
#pragma LARGE
extern void *memcpy (void *s1, void *s2, int n);

typedef unsigned char byte;

void putData (int length) reentrant {
  static xdata byte xbuffer[512];
  static xdata byte offset, cbuffer[512];
  static xdata long tmp, sfm;
  static xdata int datlength, cell;

  cell = 0;
  tmp = tmp << offset;   // wrong register used
  sfm |= tmp;

  while (length)  {
    length -= datlength;
    memcpy(xbuffer, &cbuffer[cell], datlength);
  }
}
```

☑ Corrected: printf string output with precision 0 should generate empty string

```
char unit[] = "This is a test";

void main (void) {
  unsigned int t1;

  for (t1 = 0; t1 < 20; t1++)  {
    printf("test: %.*s\n",(int)t1,unit);
  }
}
```

☑ Corrected: Following Code causes General Protection Fault

```
struct { char c1, c2; } s ;
unsigned us=&s.c2-&s.c1;
```

☑ Corrected: Following Code should generate a syntax error

```
typedef struct  {
     unsigned char Credit;
     unsigned char IndexLec;
     unsigned char IndexEcr;

} T_Type;

T_Type Variable;

void main (void)  {
  ++Variable.IndexEcr %= 3;  // these are all wrong
  ++Variable.IndexEcr /= 3;
  ++Variable.IndexEcr *= 3;
  ++Variable.IndexEcr += 3;
  ++Variable.IndexEcr -= 3;
  ++Variable.IndexEcr ^= 3;
  ++Variable.IndexEcr &= 3;
  ++Variable.IndexEcr |= 3;

  Variable.IndexEcr %= 3;     // these are Ok.
  Variable.IndexEcr /= 3;
  Variable.IndexEcr *= 3;
  Variable.IndexEcr += 3;
  Variable.IndexEcr -= 3;
  Variable.IndexEcr ^= 3;
  Variable.IndexEcr &= 3;
  Variable.IndexEcr |= 3;
}
```

☑ Corrected: Following Code should generate a ‘not a lvalue’ error

```
#define ulong unsigned long

unsigned char xdata aucIOBuf[5];
ulong ulCheckSum;

void test (void)  {
  (ulong) aucIOBuf = ulCheckSum;
}
```

☑ Corrected: local OPTIMZE values do not disable the optimization

```
#pragma ot(9,size) LARGE

unsigned char func1( unsigned char test1, unsigned char test2 )  {
  unsigned char xdata a1, a2, a3;

  a1 = test1;
  a2 = test2;
  a3 = a1+a2;

  if(test1 > test2)  a3 += test1;
  else               a3 -= test1;
  return(a3);
}


unsigned char func2( unsigned char test1, unsigned char test2 )  {
  unsigned char xdata a1, a2, a3;

  a1 = test1;
  a2 = test2;
  a3 = a1+a2;

  if(test1 > test2)  a3 += test2;
  else               a3 -= test2;
  return(a3);
}


#pragma ot(8,size)

unsigned char func3( unsigned char test1, unsigned char test2 )  {
  unsigned char xdata a1, a2, a3;

  a1 = test1;
  a2 = test2;
  a3 = a1+a2;

  if(test1 > test2)    a3 += test1;
  else                 a3 -= test2;
  return(a3);
}

#pragma ot(9,size)
unsigned char xdata x1;
unsigned char xdata x2[10];

void test () {
  func1 (x1, x2[x1]);
  func2 (x1, x2[x1]);
  func3 (x1, x2[x1]);
  func1 (x1, x2[x1]);
  func2 (x1, x2[x1]);
  func3 (x1, x2[x1]);
}
```

☑ Corrected: Compiler does not reload DPTR register with MODE2 directive

```
#pragma MODE2

extern unsigned int f(unsigned char xdata*);
typedef struct   {
  unsigned char xdata*  clazz;
```

```
  signed short    datalength;
  unsigned char xdata* datum;
} struct_0;

typedef struct  {
  struct_0  handle[1];
  struct_0  name_handle[1];
} struct_1;

typedef struct  {
  int bla1;               /* other elements */
  signed short isEnabled; /* offset to the beginning of the data struct is 4 */
  int bla2;               /* other elements */
} struct_2;


void main (void)  {
  unsigned char xdata *tmp;
  tmp = (unsigned char xdata*)f(tmp);
  tmp = ((struct_1 xdata*)(unsigned char xdata*)(tmp))->handle->datum;
  if( ((struct_2 xdata*)(unsigned char xdata*)(tmp))->isEnabled == 0 ) {
// previous line does not reload the DPTR register with the new value of tmp
    tmp = 0;
  }
}
```

☑ Corrected: Compiler does not detect that ?C?ILDIPTR call destroys R0

```
#pragma COMPACT
#include <ctype.h>

bit GetMessageCommand(unsigned char xdata *command, unsigned char xdata **buffer) {
  unsigned char pdata digit;

  digit = *((*buffer)++);  // re-use of R0, but value is destroyed
  if (isdigit(digit)) {
      *command = digit - '0';
      digit = *((*buffer)++);
    if (isdigit(digit)) {
      *command = (*command * 10) + (digit - '0');
      return 1;
    }
  }
  return 0;
}
```

☑ Corrected: Compiler reports ‘Not an lvalue’ by mistake

```
extern   unsigned char  xdata RAM_BUF[266];
unsigned int GetBlockWord(void) ;

void main (void)  {
  unsigned char length = 0;  // lokale Variable
  (unsigned int)(RAM_BUF+length)=GetBlockWord();   // not an lvalue error
  length += 2;
  RAM_BUF[1] = length-2;
  RAM_BUF[9] = 10;
}
```

☑ Corrected: Parameter overwritten due to over-optimization

```
unsigned char xdata *GetAddrA(void);
unsigned char xdata *GetAddrB(void);
```

```
void DoThis(unsigned char xdata *pIn1, unsigned int In2);

main()  {
  unsigned char xdata *pA;
  unsigned char xdata *pB;
  pA = GetAddrA();
  pB = GetAddrB();
  DoThis(pB, pB-pA-1);  // parameter 1 gets overwritten
}
```

☑ Corrected: Optimization Problem

```
unsigned long s[2][2];

void u (unsigned char i, unsigned char j, unsigned int v)   {
  s[i-1][j-1] += (unsigned long)v;
}
```

☑ Corrected: Long to 'memory specific Pointer' cast does not work

```
unsigned long address;
unsigned char *p;
#define BYTE unsigned char

unsigned int i;

void main (void)  {
  p  = (BYTE *) address;                          // cast OK
  p  = (BYTE xdata *)((unsigned int) address);  // cast should be possible
  i  = address;
  p  = (BYTE xdata *) i;
  if (((BYTE xdata *) address) == 0)  i = 1;
}
```

☑ Corrected: SRC-Mode: USING comes too late in SRC file

```
extern void test(void);
void interrupt_func(void) interrupt 3  {
  test();
}
void test(void) {
}
```

☑ Corrected: No Debugging Information for variables that are assigned to DPTR

```
void main(void)  {
    unsigned char xdata *ptr;

    ptr=(unsigned xdata *)0x0000;
    while(1)  {
        *ptr=0x00;
        ptr++;
    }
}
```

☑ Corrected: Fixup for ACALL in Common Block incorrect

```
#pragma ROM(SMALL) OT(9, SIZE)
#define uint  unsigned int
#define uchar unsigned char
```

```
#define  METER_NUM  8

#define  OFFSET_PARAM  0x30
#define  OFFSET_COUNT  0x20
#define  LENGTH_COUNT  2
#define  LENGTH_PARAM  2

extern uint ReadEEPROM (uint wOffset);
extern void WriteEEPROM (uint wData,uint wOffset);
idata uint Meter [METER_NUM];
idata uint PARAM [METER_NUM];

void main ()  {
  uchar i,mask;
  uint temp;

  for (i=METER_NUM;i;) {
    i --;
    Meter [i] =   ReadEEPROM (OFFSET_COUNT+i*LENGTH_COUNT); // wrong ACALL
    PARAM [i] = ReadEEPROM (OFFSET_PARAM+i*LENGTH_PARAM);
  }
    temp = ReadEEPROM (OFFSET_PARAM+i*LENGTH_PARAM);
}
```

☑ Corrected: Browse Info incorrect for functions with leading '_?' or '_'

```
#pragma SMALL code

long RsFunc (int i1, long l2) reentrant  {    // gives '_?RsFunc'
  l2 += i1 + 10;
  return (l2);
}

long StFunc (int i1, long l2)  {              // gives '_StFunc'
  l2 += i1 + 10;
  return (l2);
}

int  i1;
long l1;
void main (void)  {
  l1 = RsFunc (i1, l1);
  l1 = RsFunc (i1, l1);
  l1 = StFunc (i1, l1);
  l1 = StFunc (i1, l1);
}
```

☑ Corrected: Bit OR operation sometimes creates incorrect code

```
sbit KEY1=0x90;
sbit KEY2=0x91;
sbit KEY3=0x92;
sbit KEY4=0x93;

void test (void)  {
  while (KEY1 | KEY2 | KEY3 | KEY4);
}
```

☑ Still Open: Large initilizations should be constructed in two records

```
#include <stdio.h>
```

```
unsigned char array1[0x4000] = {
      0x00, 0x01, 0x02, 0x03, };
void main (void)  {
  unsigned char t;
  while (1)  {
    t = array1[1];
  }
}
```

☑ Corrected: Compiler reports 'undefined identifier' error by mistake

```
77) Undefined idendifier error given
====================================
void (*funcptr)(void) reentrant;
extern void dummy1(void) reentrant;
extern void dummy2(void) reentrant;

void test(void) {
  (funcptr = dummy1)(); // Zuweisung + Aufruf wird noch gemacht,
  funcptr();            // -> Error C202: undefined identifier
}
```

☑ Corrected: Only with MODE2: MOVB @R0 does not load the R0 register

```
#pragma MODE2
sfr SHIELD1 = 0xBE;     // SHIELD1
sfr SHIELD2 = 0xBF;     // SHIELD1

/* Copy from SFR register to idata memory */
void _copy_s2i_ (void idata * dest, unsigned char src, unsigned char len);
unsigned char idata shieldref[2];// array to store 1st and 2nd byte of SHIELD1 register

int test (void)  {
  unsigned char idata shield1data[2]; // array to store 1st and 2nd byte of SHIELD2
register
  unsigned char idata shield2data[2];    // array to store 1st and 2nd byte of SHIELD1
register

  _copy_s2i_(shield1data,SHIELD1,2);     // read 2 bytes from shield register with movb
  _copy_s2i_(shield2data,SHIELD2,2);

      if (   ((shield1data[0] ^ shield1data[1] & 0x1f) != 0x1f) // D0...D4 must be
invers of DQ0...DQ4
           || ((shield1data[0] & 0x1f) != shieldref[0])) return (1);
  return (0);
}
```

☑ Corrected: Only with MODE2: Following Code does not work

```
#pragma mode2
#define XBYTE ((unsigned char volatile xdata *) 0)
unsigned short AbsAddr=0x9000;

unsigned char xdata a[0x1000] _at_ 0;

int main() {
  return (XBYTE[AbsAddr+2]);
}
```

☑ Corrected: Following Code does not work

```
void func (unsigned int);
```

```
unsigned char xdata i;

static void vUpdateTimer (unsigned int uiTime)  compact reentrant {
  i -= uiTime;
  func(uiTime);
}
```

C51 V6.03 17. August 2000 Release Completed

Modifications and Corrections in C51 Compiler from Version 6.03 Differences to Version 6.10

Part 3: C51 Version 6.10

☑ Corrected: Parameter 'tDummy' removed by optimizer

```
// Problem with indirect function call with parameter passing with V6.03
typedef void (code *BIOS_UP)(void);
typedef void (code *TASKS_UP1)(unsigned char nTaskNr, BIOS_UP pTaskUP);
extern  code BIOS_UP pUP[10];
extern  void tDummy(void);
TASKS_UP1 fptr;

extern void check (void);

void anla (void) {
//fptr(3, tDummy);    // second parameter is not passed
  fptr(3, (void code *) tDummy);
  check ();
//((TASKS_UP1) pUP)[2](7, tDummy); // This line causes a GPF
}
```

☑ Corrected: ‘Address of’ operator ignored

```
typedef unsigned char BYTE;

struct
{
  BYTE f;
} *s;

BYTE z[10];

void main( void )  {
  BYTE x;

  x = ( ( ( BYTE idata* ) &z )[ s->f ] ); //address of ignored
  x = ( (                  z )[ s->f ] ); //ok
}
```

☑ Corrected: unused volatile loads might result in unbalanced PUSH/POP

```
//Occurs when the code contains empty if statements like:
  if (XBYTE[0x8000]) ;
```

☑ Corrected: SRC Mode generates wrong file, due to unused parameters

```
typedef unsigned char byte;
void func1(byte xdata *work,byte xdata *x,byte xdata *n,byte y)  {
    work[0] ^= x[1] ^ n[1] + y;
}
```

☑ Corrected: Wrong absolute register access generated

```
#include <philips\reg51f.h>

#define MVOLTAGE 1
#define MMILLIAMP 2
#define FALSE 0
#define TRUE 1

unsigned char mType;
idata unsigned int sampleCnt;
idata unsigned int sampleCntRel;

void avrTabInit(void) {}

void measureIsr(void) interrupt 0x1 using 2
{
  switch(mType)
  {
    case MVOLTAGE:
     {
       // mTab[mTabIdx]=getAdc(CH2);
       sampleCnt=sampleCntRel; // set sample count to startvalue
       avrTabInit(); // clear average table
     }

     case MMILLIAMP:
     {
      // mTab[mTabIdx]=getAdc(CH3);
       sampleCnt=sampleCntRel; // set sample count to startvalue
       avrTabInit(); // clear average table
     }
  } // end of switch
}

void main(void) {}
```

☑ Corrected: Cast does not change memory type anymore

```
void *l_pVar;

void test (void)  {
  l_pVar = (void xdata *) l_pVar;
}
```

☑ Corrected: Only in MODE2: Program generates MOV @DPTR+49H in E2000 mode

```
typedef struct param  {
   unsigned char  ca;
   unsigned char  cb;
   unsigned short sa;
   unsigned char  xdata * pc;
   unsigned char  cd;
   unsigned char  ce;
} tParam;
```

```
#define MAX_PARAMS_NUMBER 15

typedef struct environment  {
   unsigned char  ca;
   unsigned char  cb;
   unsigned char  cc;
   unsigned char  cd;
   unsigned char  ce;
   unsigned char  cf;
   unsigned char  cg;

   unsigned char  xdata *   p_A;
   unsigned char  xdata *   p_B;
   unsigned char  xdata *   p_C;

   unsigned char  u8_NumParams;
   tParam         params[MAX_PARAMS_NUMBER];
} tEnvironment;

tEnvironment   xdata    st_Environment;

//--------------------------------------
#define p1_ 8
#define p2_ 6
#define p3_ 2

#define DE_PTR(x) st_Environment.params[x].pc

static void ExchangeFields()
{
   st_Environment.params[p1_].pc = st_Environment.params[p2_].pc;
   st_Environment.p_B = 0;
}
```

☑ Corrected: Program generates internal error

```
char code * q ;
char c  ;

void f (char idata * p)  {
  while (p != q) {
    c = * p ;
    c ^= (char)p ;
  }
}
```

☑ Corrected: Following program is overoptimized

```
unsigned short bug1 (unsigned short toto)  {
  toto += 0x20;             // high byte not updated
  if (((unsigned char) (toto>>8)  | 0xF0 ))  toto=toto;

  return toto;
}
```

☑ Corrected: 'loc' is assigned to wrong segment, overwrites 'i'

```
#pragma code debug
void main (void)  {
    static   const int   i = 1;    // Ok.
             const int loc = 3;
```

```
  static const int far if0 = 10;
         const int far cf0 = 20;
  static const int xdata ix0 = 10;
         const int xdata cx0 = 20;
}
```

☑ Corrected: Parameter Passing is incorrect

```
#pragma code debug oe

extern unsigned char xdata dis_buffer[8][1024];
extern unsigned char code INFINEON3[];
extern void CopyCodeXdata(unsigned int, unsigned int,unsigned char) small;
#define X 204

void test(void) {
  unsigned char idata i;
  unsigned char data j;
  unsigned char data k;
  unsigned char data l;


CopyCodeXdata(&INFINEON3[(j*6*(X/12))+k*6+l*6*(X/12)*10],&dis_buffer[1][j*6+k*60+l*6*(X
/12)*10],6);
}
```

☑ Still Open: Strange C51 behaviour

```
// Why is using of '(' not signaled as an error ?
// The generated code does not initialize the table correctly !
//
char strange_table_init[] = ( 1,2,3,4,5,6,7,8 );
//--------------------------^ this is legal C !!! (result is 8)
char dummy1, dummy2;

// not all dead code eliminated
void main()   {
  while ( 1 );           // Loop forever ... but (look down)

  if ( dummy1 == 0x12 )  // Any value ... it does not matter
    dummy2 = 0x34;
  else
    dummy2 = 0x56;       // This line (after IF's else gets compiled
}
```

☑ Corrected: ROM(SMALL) OT(9,SIZE): AJMP-Ins misses Fixup

C51 V6.10 7. November 2000 Release Completed

Modifications and Corrections in C51 Compiler from Version 6.10 Differences to Version 6.11

Part 4: C51 Version 6.10

☑ Corrected: strrpbrk function did not work correct

☑ Corrected: More than 18 include files may cause General Protection Fault

---

☑ Corrected: Complex constant pointer operations may cause wrong code

```
Example statement:
   timer = ((hardware_timer_t xdata *)&micro.hw_timer1_m1)->timer_list[index];
```

☑ Corrected: Compiler generates invalid CJNE DPH/DPL,#const instructions

```
#define C1 ((char code *)0x1FF4)
extern char idata B1[128];

void Bug(void)
{
 char idata *s;
 char code *p;

 s = B1;
 p = C1;
 do
  {
   *s = *p;
   p++;
   s++;
  } while (p != C1 + 4);
}
```

☑ Corrected: Cannot Optimize Function Error with following code

```
typedef struct
{
      char VarMember4 ;
      char VarMember5 ;
      char VarMember6 ;
      char VarMember7 ;
      char VarMember11 ;
} StructType ;

void FUNC_A (char, char, char, char) ;
void FUNC_B (StructType*, char) ;

void FUNC_B (StructType* VarParamF, char VarParamG)
{
      FUNC_A (VarParamF->VarMember4, VarParamF->VarMember5,
            VarParamG, VarParamF->VarMember11) ;
}
```

☑ Enhancement: Maximum Number of Local Labels increased to 8192

☑ Corrected: Code generates Fixup Error at Linker step

```
#define  AnzMpKnoten                  8     // Anzahl MP-Knoten pro Bus

unsigned char xdata         ReadMP1[50];
signed int    xdata         rdat[200];

LadeMpIstwerte () {
  unsigned int  xdata *IstWord  = &ReadMP1[AnzMpKnoten - 1]; // Index[1..] adressiert
  unsigned char xdata *IstByte1 = &ReadMP1[3 * AnzMpKnoten]; // Index[1..] adressiert
  unsigned char xdata *IstByte2 = &ReadMP1[4 * AnzMpKnoten]; // Index[1..] adressiert

  rdat[1] = IstByte1[1];
  rdat[2] = IstByte1[2];
  rdat[3] = IstByte1[3];
  rdat[4] = IstWord[2];  // Over Optimized
}
```

☑ Corrected: Incorrect short branches with MODE2 and O2

☑ Corrected: Address Sync Error (caused by MOV DPTR,#LOW word)

```
#pragma MODE2
#define SLE66CX320P
#include <reg66cx.h>
#include <intre2.h>
typedef unsigned char Byte;
void test()
{
    Byte work[16+2];
    Byte i;
    for (i = 0; i < sizeof(work-2); i += 1) work[i] = i;
    CRCC = 0x03;
    CRCC = 0x00;
    {
//#define ATLEAST_COMPILE
#ifdef ATLEAST_COMPILE
        void xdata *p = work; /* without this compiler barfs, unfortunately
                                 it allocates memory (stupid compiler) */
#endif
        _copy_x2s_(CRCD,(void xdata *)work,16);
    }
    work[16+0] = CRCD;
    work[16+1] = CRCD;
}

void main(void)  {
  test();
  while (1) ;
}
```

☑ Corrected: cror/crol may cause internal errors

```
#include <intrins.h>
typedef void (*pFn)(void);
unsigned char ins;

void dummy_routine(void);

void dispatch (void) {
```

```
pFn const ptr2func[2] = {dummy_routine, dummy_routine};
 (*ptr2func[_cror_((unsigned char)(ins-0x12),1)]) ();
}
```

C51 V6.11 23. January 2001 Release Completed

Modifications and Corrections in C51 Compiler from Version 6.11 Differences to Version 6.12

Part 5: C51 Version 6.12

☑ Corrected: typedef with function pointer did not take the reentrant attribute

```
typedef void DTD_Func (void *p1, void *p2) reentrant;
DTD_Func *DTD_DE;     // reentrant attribute  missing
// without typedef it is OK
void (*DTD_DE2) (void *p1, void *p2) reentrant; // works OK

void main (void)  {
  DTD_DE = ((void *) 0);
  DTD_DE  (((void *) 0), ((void *) 0));
  DTD_DE2 (((void *) 0), ((void *) 0));
}
```

☑ Corrected: Parameter that is used twice is overwritten in following example

```
struct m {
  unsigned char  cobj;
  unsigned int     id;
  unsigned int   attr;
  unsigned int    len;
};

struct m M[10];

extern void setCobId (unsigned char mt, unsigned long id) small;

static void updateCobId (unsigned char mt) {
  setCobId(mt, M[mt].id);     // mt is not correctly passed to setCobId
}
```

☑ Corrected: Incorrect code

```
long xdata larr[100];
long * p = &larr[10];
long **pp;
unsigned int ui = 20;

void main (void) {
//pp = &p;
   *pp += 0-ui;          //    *pp = x:0028   error !!!!!!
//*pp += ui;            //    *pp = x:0028   error !!!!!!
  *pp = *pp -ui;         //    *pp = x:0028   error !!!!!!
//*pp = *pp - ui;        //    *pp = x:0028   error !!!!!!
//*pp = *pp + ui;        //->  *pp = x:0078   ok
  *pp = *pp - 20;        //->  *pp = x:0028   ok
//while (1);
}
```

☑ Corrected: 'void *' + something does not cause an error

```
void main (void)  {
  void *p1;
  char  a1;

  p1 += a1;
}
```

☑ Corrected: Initialization Problem with O2 and 'const xdata'

```
#pragma MODE2 OMF251 VARBANKING
const unsigned char xdata xc[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
void main (void)  {
  unsigned char vc[10] = {12,11,10,9,8,7,6,5,4,3};
  vc[0] = xc[0];
}
```

☑ Corrected: OMF2: Symbols truncated at 40 characters

☑ Corrected: 'const' Problem

```
//#pragma USERCLASS (CONST = "MyConst")  // whis works.
char code abc = 1;  // should be in class CONST but is in CODE.
```

☑ Corrected: Warning/Error is missing

```
extern char ar[26];
char ar[25];   // should give error/warning
```

☑ Corrected: Error for the sbit definition is missing

```
struct  SMKEYBITMAP {
   unsigned btTranCmdEncKey  : 1;
   unsigned btTranRespEncKey : 1;
   unsigned btTranCmdMacKey  : 1;
   unsigned btTranRespMacKey : 1;
   unsigned btCmdEncKey      : 1;
   unsigned btRespEncKey     : 1;
   unsigned btCmdMacKey      : 1;
   unsigned btRespMacKey     : 1;
};

union A {
  struct SMKEYBITMAP ucSMKeyBitmap;
  char a;
};

union A bdata x;

sbit btTranCmdEncKeyA = x.ucSMKeyBitmap.btTranCmdEncKey;
sbit btTest = x.a^1;

void test(void){
   x.a = 0;
```

```
   x.ucSMKeyBitmap.btTranCmdEncKey = 0;
   btTranCmdEncKeyA = 0;  // <-- Hier verschluckt sich der C-Compiler
   btTest = 1;
}
```

☑ Corrected: Following Code did not reload the Pointer

```
unsigned int data  *p1;
unsigned int idata iv1;
unsigned int idata iv2;

void test (void)  {
  p1  = &iv1;
  iv2 = 200;
  *p1 = iv2;
  *p1 += 1;         // here R0 is not reload
  *p1 = *p1 + *p1; // generates an error in this line
//*p1 = *p1 + 100; // does not work either
  iv1 = *p1;
}
```

☑ Corrected: General Protection Error on following code

```
#include <reg51.h>

unsigned char bCode;

void main(void)  {
  switch (bCode)
  {
    case SELECT_FIRST:      f
    case SELECT_NEXT:       break;

    default:
      break;
  }
}
```

☑ Corrected: General Protection Error on following code because of a syntax error

```
#include <stdio.h>

char tmp;
void main (void)  {
      printf("Some string\n")             //<- missing semicolon
      (tmp == 0) ? (tmp = 1) : (tmp = -1);
}
```

☑ Corrected: Incorrect code

```
#define UCHAR unsigned char
#define UINT  unsigned int

typedef struct
    {
    UCHAR Pin[4];
    UCHAR Zeitzonen[4];
    UINT  Kto_Nr;
    } Benutzerkonto;

data UCHAR                zahl;
UINT xdata                hilfswort;
```

```
data long                   longzahl;

UCHAR iy;

UCHAR bug1(UCHAR xdata *zif, UCHAR ix) {
//  zif += ix;
    ix = (*(zif += ix) & 15) * 10;    // ix not cast to uint.
//  ix = (*(zif += ix));
    return ix;
}
```

☑ Corrected: O2 directive cannot be given in #pragma lines

```
#pragma O2

int far x;   // generates syntax error, but should be
             // enabled with the O2 directive
```

☑ Corrected: In Dallas 390 Contigiuous Mode, no OPTR library calls can be done

```
char *strncpy (char *s1, int n)  {
  s1[n] = 0;
}
```

☑ Corrected: Only with MODE2: long store via xdata pointer with offset is wrong

```
#pragma PAGEWIDTH(110)
#pragma PAGELENGTH(100)
#pragma MODE2

typedef struct{
  unsigned char uc[5];
  unsigned short us;
  unsigned long ul;
  signed long sl;
}STR;

STR xdata vargl0000[3];
STR xdata * xd_str_p;

void main(void){
    /* hier gehts daneben */
  xd_str_p->ul = 1;
  xd_str_p->us += 5;
  xd_str_p->ul += 5;
  while(1);
}
```

☑ Corrected: Initialization failure

```
char test (char c1, char c2)  {
  return (c1 | c2);
}
      char code szB1[] = "Test";
const char  far szB2[] = "Test";  // Initl: wrong segment ?CO?xx
```

☑ Corrected: SRC-Mode: missing or duplicate symbols

C51 V6.12 09. March 2001 Release Completed

Modifications and Corrections in C51 Compiler from Version 6.12 Differences to Version 6.1x

Part 5: C51 Version 6.1x

☑ Corrected: Only with MODE2: following code generates incorrect access

```
#pragma MODE2

typedef struct {
  long min;
  long max;
} Range;

Range xdata *target;

long f (void) {
  long max,min;

  min = target->min;
  max = target->max;   // offset incorrect
  return(max-min);
}
```

☑ Corrected: strtoul does not work