Keil C51 Bugfix Report

Modifications, Enhancements and Corrections in Keil C51 Compiler Version 6

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*6
0+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

Part 4: C51 Version 6.11

☑ 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

☑ 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

Part 6: C51 Version 6.14

☑ Corrected: following code does not work

```
#pragma O2

unsigned char  far hd_uc0;
unsigned short far hd_us0;
unsigned char  far * far hd_p_hd_uc0[2];
unsigned short far * far hd_p_hd_us0;

unsigned char i;

void main(void){
      hd_p_hd_uc0[i] = &hd_uc0;
      hd_p_hd_us0 = &hd_us0;    // overwrites R1-R3
      while(1);
}
```

☑ Corrected: following code is broken since C51 V6.11

```
#pragma LARGE
typedef unsigned int UINT;
typedef unsigned char UCHAR;
UINT uiSscRecBuf [100];

static UCHAR GetRxByte (UINT uiBitOfs)  {
  UINT uiVal;
  UCHAR ucShift = uiBitOfs & 7;
/*  the following statement is the broken statement: */
  uiVal = *(UINT xdata*) (&((UCHAR xdata*)&uiSscRecBuf) [uiBitOfs >> 3]);
  uiVal <<= ucShift;
  return ( *(UCHAR xdata*)&uiVal );/* oberes wort zuruckgeben */
}
```

☑ Corrected: 'NOEXTEND' control ignored

```
#pragma NOEXTEND
int   data;
int   xdata;
void main (void)  {
  data = 1;
  xdata = 2;
}
```

☑ Enhancement: Improvement on register assignment

☑ Enhancement: Redundant Load Optimization Improved

```
#pragma LARGE

unsigned int ui_FM_Get_Entry (unsigned char code *pucDFName, unsigned char code
*pucEntry);

bit b_FM_Delete_Entry (unsigned char code *pucDFName, unsigned char code
*pucEntry)  {
  unsigned int uiEntryNr;
  uiEntryNr = ui_FM_Get_Entry (pucDFName, pucEntry);  // reloads pucDFName
  if (uiEntryNr == 0)  return 0;
  pucEntry = 0;
  return (0);
}
```

☑ Corrected: Incorrect OMF Record EXTDEF with directive O2

```
#pragma O2 COMPACT
extern void func (long a, long b, long c);
void main (void)  {
  func (1,2,3);
}
```

☑ Enhancement: #warning added

```
int x1;
int x2;
#warning "BullWarning"
void main (void)  {
  x1 = 1;
  x2 = 2;
}
```

☑ Corrected: DF(id) on the command line sould be '#define id', is '#define id 1'

☑ Enhancement: Number of Possible Int Register Variables increased

☑ Corrected: ROM(D512K)/ROM(D16M) did not set ModInf in OMF file

☑ Corrected: SRC-Mode: missing data definitions

```
#pragma SB DB OE MODDP2 CD LA OT(9,SPEED) SRC

extern bit     fSendI2CEvent;               // this 'extern' causes
bit        fSendI2CEvent = 0;               // undefined id in Src file
extern unsigned char xdata bI2cRxTaskSlot;  // this 'extern' causes
unsigned char xdata bI2cRxTaskSlot = 0;     // undefined id in Src file

void main (void)  {
}
```

☑ Corrected: 'far' access with O2 added

```
#pragma O2 ROM(D512K)

#define FVAR(object, addr)   (*((object volatile far *) (addr)))
#define FARRAY(object, base) ((object volatile far *) (base))

unsigned char  far  uc1[10];
unsigned long  far  ul1[10];
unsigned char  far  uc2;

void main (void)  {
  FVAR (char, 0x095678) = 0;
  FVAR (char, 0xC25678) = 0;
  *FARRAY (char, uc1) = 10;
  *FARRAY (long, &ul1[2]) = 0x12345678;
  while (1);
}
```

☑ Corrected: 'short const xdata id' not allocated to XCONST

```
#pragma MODE2 XCROM O2 DB

//#pragma UCL(code = EEPROM)
//#pragma userclass(xdata = APP2)
//#pragma USERCLASS(xconst = EEPROM)
//#pragma ucl(hconst = EEPROM)

short       xdata xd_ss1;
short const xdata xc_ss1 = 11;   // --> not allocated to XCONST !
const short xdata xc_ss2 = 11;   // Ok.
short const far   hc_ss1 = 12;

void app2(void){
//    xd_ss1 = xc_ss1;
//    xd_ss1 = hc_ss1;
}
```

☑ Corrected: SRC-Mode: incorrect bit-seg locals

```
#define BYTE  unsigned char
#define BOOL  bit

BOOL s_is_prime (BYTE p_addr, BYTE p_length,
                 BYTE r_addr, BYTE r_length, BOOL Bp1, BOOL Bp2)  {
  BYTE x_addr;
  BYTE dp_addr;
  BYTE dp_len;
  BOOL ret_val;

  x_addr = p_addr | r_length;
  dp_addr = p_length;
  dp_len  = r_addr;
  if (dp_addr > (x_addr + dp_len))  {
    ret_val = 1;
  }
  else ret_val = 0;
  ret_val |= Bp1 | Bp2;
  return (ret_val);
}
```

☑ Enhancement: More Register Variables used

```
extern char c_3Param(int iA, int iB, int iC);

int v_test5(int iA, int iB, int iC) {  // iB wird abgespeichert
  int iD, iE;
  iE = c_3Param(iA, iC, iB);
  iD = 0xff;
  return (iD + iE);
}
```

☑ Enhancement: NOLINES/LINES control added

```
long l1;

#pragma save nolines
void Test1 (int i1, int i2)  {
  l1 = i1 * i2;
}
```

```
#pragma restore
void Test2 (int i1, int i2)  {
  l1 = i1 * i2;
}
```

☑ Corrected: following code removes memory increment

```
unsigned char data x = 1;
unsigned char data * data y = &x;

void test2(void)  {
  x  = ++x;     // memory increment is removed
}

void test(void)  {
  *y  = ++*y;   // memory increment OK
}
```

☑ Corrected: following code generates wrong type for address index calculation

```
unsigned char u_global;
signed char   s_global;
extern xdata short array[10];

short err1(void) reentrant {
  /*----------------------*/
  unsigned char local;

  local = u_global;
  return array[local]; /* uses u_global as unsigned short! */
}

short okay1(void) { /* not reentrant */
  /*-------------*/
  unsigned char local;

  local = u_global;
  return array[local];
}

short okay2(void) reentrant { /* no local */
  /*-----------------------*/
  return array[u_global];
}

short okay3(void) reentrant { /* signed global */
  /*-----------------------*/
  signed char local;

  local = s_global;
  return array[local];
}

short okay4(void) reentrant { /* casted index */
  /*-----------------------*/
  unsigned char local;

  local = u_global;
  return array[(unsigned int)(local)];
}
```

☑ Enhancement: Common Code Optimizer does not detect ?BYTE symbol sequences

```
double dpowu
 (double d, unsigned u)
 {
 double p = 1.0;

 while (u-- > 0)
 p *= d;
 return p;
}

double Delta, LDelta, FDelta;

test () {
 Delta = 0.5 / dpowu(10.0, (6 - 1) - 1);
}

Ltest () {
 LDelta = 0.5L / dpowu(10.0, (6 - 1) - 1);
}

Ftest () {
  FDelta = 0.5 / dpowu(10.0, (6 - 1) - 1);
}
```

☑ Corrected: Incorrect Fixup generated (with OMF2 directive)

```
#pragma code O2
char iFunc (char c1, char c2)  {
  return (c1 | c2 | 0x33);
}
void test (void)  {
  iFunc (0x08, 0x20);
  ((void (code *) (void)) 0x0000)();   // incorrect Fixup here.
  iFunc (0x08, 0x20);
}
```

☑ Corrected: Pointer Cast might yield to wrong code

```
char c;

void main(void)  {
  unsigned long origen;
  unsigned char xdata z[128];

  origen = 2;                        // Address of origin

  for (c = 0; c < 3; c++)  {
    z[c] = c;
  }

  if(origen == (unsigned long)z)    {
    c = 0;
  }
}
CORR: ASMGEN.C(lcp) /20.6.2001/
```

☑ Corrected: switch/case has potential problems when there are exact 256 cases

```
#pragma mode2

void main(void){
      unsigned char uc_tmp;
      while(1){
            uc_tmp = 223;
            switch(uc_tmp){
                  case 0:
                  case 1:
                  case 2:
                  case 3:
                  case 4:
                  case 5:
                  case 6:
                  case 7:
                  case 8:
                  case 9:
                        uc_tmp += 1;
                  case 10:
                  case 11:
                  case 12:
                  case 13:
                  case 14:
                  case 15:
                  case 16:
                  case 17:
                  case 18:
                  case 19:
                        uc_tmp += 2;
                  case 20:
                  case 21:
                  case 22:
                  case 23:
                  case 24:
                  case 25:
                  case 26:
                  case 27:
                  case 28:
                  case 29:
                        uc_tmp += 3;
                  case 30:
                  case 31:
                  case 32:
                  case 33:
                  case 34:
                  case 35:
                  case 36:
                  case 37:
                  case 38:
                  case 39:
                        uc_tmp += 4;
                  case 40:
                  case 41:
                  case 42:
                  case 43:
                  case 44:
                  case 45:
                  case 46:
                  case 47:
                  case 48:
                  case 49:
                        uc_tmp += 5;
                  case 50:
```

```
case 51:
case 52:
case 53:
case 54:
case 55:
case 56:
case 57:
case 58:
case 59:
      uc_tmp += 6;
case 60:
case 61:
case 62:
case 63:
case 64:
case 65:
case 66:
case 67:
case 68:
case 69:
      uc_tmp += 7;
case 70:
case 71:
case 72:
case 73:
case 74:
case 75:
case 76:
case 77:
case 78:
case 79:
      uc_tmp += 8;
case 80:
case 81:
case 82:
case 83:
case 84:
case 85:
case 86:
case 87:
case 88:
case 89:
      uc_tmp += 9;
case 90:
case 91:
case 92:
case 93:
case 94:
case 95:
case 96:
case 97:
case 98:
case 99:
      uc_tmp += 10;
case 100:
case 101:
case 102:
case 103:
case 104:
case 105:
case 106:
case 107:
case 108:
case 109:
      uc_tmp += 11;
case 110:
```

```
                case 111:
                case 112:
                case 113:
                case 114:
                case 115:
                case 116:
                case 117:
                case 118:
                case 119:
                      uc_tmp += 12;
                case 120:
                case 121:
                case 122:
                case 123:
                case 124:
                case 125:
                case 126:
                case 127:
                case 128:
                case 129:
                      uc_tmp += 13;
                case 130:
                case 131:
                case 132:
                case 133:
                case 134:
                case 135:
                case 136:
                case 137:
                case 138:
                case 139:
                      uc_tmp += 14;
                case 140:
                case 141:
                case 142:
                case 143:
                case 144:
                case 145:
                case 146:
                case 147:
                case 148:
                case 149:
                      uc_tmp += 15;
                case 150:
                case 151:
                case 152:
                case 153:
                case 154:
                case 155:
                case 156:
                case 157:
                case 158:
                case 159:
                      uc_tmp += 16;
                case 160:
                case 161:
                case 162:
                case 163:
                case 164:
                case 165:
                case 166:
                case 167:
                case 168:
                case 169:
                      uc_tmp += 17;
                case 170:
```

```
case 171:
case 172:
case 173:
case 174:
case 175:
case 176:
case 177:
case 178:
case 179:
      uc_tmp += 18;
case 180:
case 181:
case 182:
case 183:
case 184:
case 185:
case 186:
case 187:
case 188:
case 189:
      uc_tmp += 19;
case 190:
case 191:
case 192:
case 193:
case 194:
case 195:
case 196:
case 197:
case 198:
case 199:
      uc_tmp += 20;
case 200:
case 201:
case 202:
case 203:
case 204:
case 205:
case 206:
case 207:
case 208:
case 209:
      uc_tmp += 21;
case 210:
case 211:
case 212:
case 213:
case 214:
case 215:
case 216:
case 217:
case 218:
case 219:
      uc_tmp += 22;
case 220:
case 221:
case 222:
case 223:
case 224:
case 225:
case 226:
case 227:
case 228:
case 229:
      uc_tmp += 23;
case 230:
```

```
                  case 231:
                  case 232:
                  case 233:
                  case 234:
                  case 235:
                  case 236:
                  case 237:
                  case 238:
                  case 239:
                        uc_tmp += 24;
                  case 240:
                  case 241:
                  case 242:
                  case 243:
                  case 244:
                  case 245:
                  case 246:
                  case 247:
                  case 248:
                  case 249:
                        uc_tmp += 25;
                  case 250:
                  case 251:
                  case 252:
                  case 253:
                  case 254:
                  case 255:
                        uc_tmp *= 7;
                  break;
            }

//    while(1);
      }
}
```

☑ Corrected: ASMPARM: Problem with parameter register order

```
#pragma mode2
#pragma asmparms

extern void func(unsigned char* p, unsigned char* q, unsigned char* s);


main1()
{
      unsigned char buf1[10];
      unsigned char buf2[10];
      unsigned char buf3[10];

      func(buf1, buf2, buf3);
}

#pragma OT(4)

main2()
{
      unsigned char buf1[10];
      unsigned char buf2[10];
      unsigned char buf3[10];

      func(buf1, buf2, buf3);
}
```

☑ Corrected: Cast on left side causes problems

```
// regvars: high part gets incremented.  Customer expects low part
//                       -Endian-    -----^ this was NEVER true !!!!
#if 1
void effect1 (void)  {
 unsigned int x;

 x = 1;
// *((unsigned char *) &x) = 0;
 (unsigned char) x -= 1;  // identical to *((unsigned char *) &x) -= 1;

 if ((unsigned char) x != 0) goto error;
 return;

error:
  x++;
  goto error;
}
#endif


#if 1
// on DPTR: generates internal errors
sfr16  DPTR = 0x82;
void fff() {
  DPTR++;                      // correct, but inefficient.
  (*(char xdata *) DPTR)++;    // generates now 'not an lvalue' !
  (*(int  xdata *) DPTR)++;    // generates now 'not an lvalue' !
  ((char  xdata *) DPTR)++;    // generates now 'not an lvalue' !
  ((int   xdata *) DPTR)++;    // generates now 'not an lvalue' !
}
#endif

#if 1
// with memory typed pointers -> generic pointers are assumed
unsigned int P_;
void f2()  {
  ((char data  *)P_)++;    // gives error now.
  ((int xdata  *)P_)++;    // gives error now.
}
#endif
```

☑ Corrected: No Warning given

```
extern void test (char xdata *xp1);
char  c1;
char xdata *xp2;

void main (void)  {
  test (c1);       // should give a Warning
  test (xp2);
}
CORR: Cexpr.C (added WarnParg /6.7.2001/)
// Wieder totgelegt ! /9.7.2001/
```

☑ Corrected: Compiler uses wrong register in very complex application

```
Test-File: \C51TEST\V6BUGS\B157.C
```

☑ Corrected: PUSH/POP wrong with VARBANKING(1) directive

```
#pragma VARBANKING (1)
#include <stdio.h>                        // prototype declarations for I/O
functions
#include <absacc.h>
/*
 * this is an large array stored in extended 16MB xdata memory of the ADuC812
 */
unsigned char far large_array[0xC000];

/*
 * the following timer interrupt routine uses a variable in xdata space
 */
#define PERIOD     -250                    // 250 usec interrupt period
unsigned char xdata timer_tick;            // xdata variable

void timer0 (void) interrupt 1 using 1 {   // Int Vector at 000BH, Reg Bank 1
  timer_tick++;                            // increment xdata variable
}
```

☑ Corrected: Cannot Optimize Function Error

```
unsigned char intstr[8];
void transmit_string(char *s, char mode);

void rs_kompatible_prozent_senden(char *antwort, int value) {
 intstr[0] = (value % 1000 / 100)+48;
 intstr[2] = value % 10 +48;
 transmit_string(antwort,0);
}
```

☑ Corrected: Incorrect Warning given (single line comment...)

```
// text comment \

#define xx \
  1 \
 +2

char c1 = xx;
```

☑ Corrected: Problem with Error Message

```
sbit yy=IP^0;     // Error Message OK!

struct x {
  unsigned char y[5];
} x;

unsigned char abc;
void test (void)  {
  x.y[abc] = 8;   // there should be no error message for this line!
}
```

☑ Corrected: Optimize(9): Using not assembled correctly

☑ Corrected: Order Problem with Optimizers when OMF2 format used

---

☑ Corrected: ?DT?v_BitParmLast segment not generated with SRC

```
#define uint  unsigned int
#define uchar unsigned char

void v_BitParmFirst(bit btSet, uint uiD,uchar ucE)  {  // Ok.
  uiD = ucE;
}
void v_BitParmLast(uint uiA, uint uiB, bit btQuery)  { // ?DT? seg missing
  uiA = uiB;
}

void main()  {
  v_BitParmFirst (1, 0x1234, 'e');
  v_BitParmLast (0x3422, 0x62 /*, 0*/);
}
```

☑ Corrected: CPL bit Instructions are optimized away

```
bit   testbit;

void z1(void)  {
  testbit = ~testbit;  // CPL bit missing
}

void z2(void)  {
  if (testbit)   {
    testbit=0;
  }
  else  {
    testbit=1;
  }
}

void z3(void)  {
   testbit=!testbit;   // CPL bit missing
}
NOTE: Problem was introduced with a new optimizer in C51 V6.14c
```

☑ Corrected: #Low in assembler output missing when fancy pdata cast's used

```
#define u8 unsigned char

extern u8 xdata SK_MAK[];

typedef struct  {
  u8 b[8];
} u8_blk;

typedef struct  {
  u8_blk   l;
  u8_blk   h;
} u16_blk;
```

```
typedef struct {
  u16_blk key;
  u8      e;
  u8      a;
} DK;

DK pdata *t (void)  {
  return ((DK pdata *) SK_MAK)->e;  // MOV R0,#SK_MAK+010H ;LOW prefix missign
}
```

☑ Corrected: Unnecessary Warning given

```
typedef unsigned int size_t;
#define offsetof(s,m)   (size_t)&(((s *)0)->m)

unsigned char d, e, f;
void test (void)  {
  typedef struct {
    char b, c[5];
  } TStruct;
//unsigned char d, e;

  e = 2;
/* In den beiden folgenden Zeilen wird d auf 3 gesetzt */
  d = offsetof (TStruct, c[2]); // Kein Warning
  f = offsetof (TStruct, c[e]); // Warning C196
}
```

☑ Corrected: ARx Symbols in ECO 2000 Instructions always use the last using attribute

```
#pragma MODE2

long xdata current;
long xdata c1,c2,c3,c4;

void test (void)  using 0  {
// c1 = c2 * c3 + c4 * c1;
 current++ ;
}
void timint (void) using 1 {
  current++;
}
void test2 (void)  using 2  {
 current++ ;
}
```

☑ Corrected: #HIGH/#LOW missing

```
#pragma OPTIMIZE (9,SIZE) BROWSE cd ob pl(30000) mode2 DEBUG OBJECTEXTEND
extern unsigned char xdata chaining_buf[];
typedef struct
{
    unsigned char xdata * c;
    unsigned char xdata * d;
} ScanResult;

bit TBX_TLV_Cmp_Headerlist(unsigned char xdata* IF_Start,
                                    unsigned int IF_len,
                           unsigned char xdata* HLDO_Start,
                           unsigned char xdata* HLvalue_Start);
```

```
void testfunc (void) compact
{
  unsigned char xdata* Cert_HL_Adr;

      ScanResult xdata *ptr_scanresult;


  TBX_TLV_Cmp_Headerlist(
                   (ptr_scanresult->d)+1,
                          *(ptr_scanresult->d),
                          Cert_HL_Adr,

                        chaining_buf+2); // Fehler

/* so wuerde der Fehler nicht auftreten
   TBX_TLV_Cmp_Headerlist(0,0,0, chaining_buf+2); // Fehler tritt nicht auf
*/
}
/*--- another example ---*/
typedef bit BOOL ;
typedef unsigned char CBOOL ;
typedef unsigned char UCHAR ;
typedef unsigned int UINT ;



static CBOOL xdata l_CurModeVar ;
static UINT xdata l_SeringAff ;
static UINT xdata l_TabFleche [10] ;
static UINT xdata l_LimCur ;
static UCHAR xdata l_LimStat ;
static UINT xdata l_LimSeuilL ;
static UINT xdata l_LimSeuilM ;
static UINT xdata l_LimSeuilH ;
static UINT xdata l_LimMecaCur ;
static CBOOL xdata l_LimMeca ;


static void pres_UpdateAl( void ) ;
static void pres_UpdatePreAl( BOOL NeedUpd ) ;

static UINT pres_GetLimSeuilUtil( UCHAR data * LimStat, BOOL Real ) ;
static UINT pres_GetLimVarUtil( UCHAR data * LimStat, BOOL Real ) ;
static void pres_UpdateFleche( UINT Pres, BOOL NeedUpd ) ;

static void pres_PrepTabFleche3Niv( UINT LimH, UINT LimM, UINT LimL,
                                    UINT xdata * Tab ) ;
static void pres_PrepTabFlecheVar( UINT Limite, UINT xdata * Tab ) ;



static void pres_UpdateLimite( void )
{
   UCHAR LimStat ;
   UINT Limite ;

   if ( l_LimMeca )
   {
      LimStat = 4 ;
      Limite = l_LimMecaCur ;
   }
   else
   {
      if ( l_CurModeVar )
         Limite = pres_GetLimVarUtil( &LimStat, 0 ) ;
      else
```

```
         Limite = pres_GetLimSeuilUtil( &LimStat, 0 ) ;
   }

   l_LimCur = Limite ;
   l_LimStat = LimStat ;

   if ( l_CurModeVar )
      pres_PrepTabFlecheVar( Limite, l_TabFleche ) ;
   else
   {
      if ( LimStat == 4 )
         pres_PrepTabFleche3Niv( Limite, 0, 0, l_TabFleche ) ;
      else
         pres_PrepTabFleche3Niv( l_LimSeuilH, l_LimSeuilM, l_LimSeuilL,
l_TabFleche ) ;
   }

   pres_UpdateFleche( l_SeringAff, 1 ) ;
   pres_UpdateAl() ;
   pres_UpdatePreAl( 1 ) ;
}
```

Part 7: C51 Version 6.20

☑ Corrected: Wrong usage of registers

```
typedef unsigned char UCHAR ;
typedef unsigned int UINT ;

typedef UCHAR (code * f_FoncParIM)( UCHAR Mode, UINT Param, void pdata * Buf )
;

typedef struct
{
   UCHAR code * Form ;
   UCHAR Size ;
   UINT Param ;
   f_FoncParIM Func ;
} s_DescIdParIM ;

static UCHAR cpim_CallFunc( UCHAR Mode, s_DescIdParIM code * pdipi,
                                        void pdata * Buf )
{
   return (* pdipi->Func)( Mode, pdipi->Param, Buf ) ;
}
```

☑ Corrected: Code Improvement (More Register Variables, Return of int 0)

```
int func (int i)  {
  if (i > 0x50)  return (i+1);
  else           return (i);
}

char c;
int k;
int func2 (int i)  {
  switch (c)  {
    case 0:
      k = i+1;
      break;

    case 1:
```

```
      k = i;
      break;

    case 2:
      return (i+1);

    case 3:
      k = i*2;
      break;
  }
  return (0);
}
```

☑ Corrected: ROM(D512K) and SRC mode - incorrect branches

```
#pragma ROM(D512K)

// $MOD_CONT fehlt um DS390 contigious mode zu wahlen.
// Alle LJMP/LCALL Befehle sollten AJMP/ACALL sein.
//   der JZ  wird mit $+5 uebersetzt, aber LJMP ist in
//   diesem Mode bereits 4 byte lang.

sfr  AUXR1 = 0xA2;
sfr  AUXR  = 0x8E;
sfr  SP    = 0x81;
sfr  ACC   = 0xE0;
sfr  DPL   = 0x82;
sfr  DPH   = 0x83;
sfr  DPX   = 0xE1;

int i;
char c;
unsigned char xdata xar[0x1000];
unsigned char code  car[0x1000][2];
unsigned char xdata *xp;
unsigned char code  *cp;

int func (void)  {
  while (1)  {
    c = cp[c];
    c = xp[i];
    i = xar[i];
    if (i)  {
      c = car[i][c];
      c = car[i][c];
      c = car[i][c];
      c = car[i][c];
      c = car[i][c];
    }
    c = car[i][c];
  }
}
```

☑ Enhancement: Modifications for XDATA/CODE Segments other then 0:xxxx

```
#pragma ROM(D512K)
int i;
char c;
unsigned char xdata xar[0x1000];
unsigned char code  car[0x1000][5];
unsigned char xdata *xp;
unsigned char code  *cp;
```

```
int func (void)  {
  while (1)  {
    c = cp[c];
//  c = xp[i];
    i = xar[i];
    if (i)  {
      c = car[*xp][*cp];
      c = car[i][c];
      c = car[i][c];
      c = car[i][c];
      c = car[i][c];
    }
    c = car[i][c];
  }
}
```

☑ Corrected: SRC file contains a wrong segment in following code

```
#pragma SRC
unsigned int ui_MixedParms(unsigned char ucA,bit btCheck, unsigned int uiB){
  if (btCheck)
    return(ucA + uiB);
  return(uiB - ucA);
}
```

☑ Corrected: Following program overwrite register R1

```
unsigned char bdata control_table[8] _at_ 0x20;
#define BURST_PULSE_COUNT   (control_table[2] >> 4)
unsigned int  idata    differ_values[4];
extern void set_tx_power (void);
volatile unsigned char task_command;
#define LBO_REQUEST          0x20                    // L.B.O task
volatile bit shot_flag;

unsigned char info_byte (void) {
      unsigned int      xi, yi;
      unsigned char     tmp;        //

      tmp = 0xFF;
      if (task_command & LBO_REQUEST)
            tmp &= 0xFD;
      if ( ( (differ_values[0] + differ_values[2]) < (BURST_PULSE_COUNT+1) *
1200 ) ||           // min = 600 * 2
             ( (differ_values[1] + differ_values[3]) < (BURST_PULSE_COUNT+1) *
1200 ) )
            tmp &= 0xEF;
      xi = (differ_values[0] + differ_values[2]) >> 3;
      yi = (differ_values[1] + differ_values[3]) >> 3;
      if ( (3 * xi > 4 * yi) || (3 * yi > 4 * xi) )
            tmp &= 0xFF;
              // ??? channel unbalance
      if (!shot_flag)
            tmp &= 0xFE;
      return (tmp);
}


void adjust_power (unsigned char pow_lvl) {
      unsigned int      dual;
      unsigned char     power, level;

      power = pow_lvl >>  4;                                          // power
```

```
      level = pow_lvl & 0x0F;
      if ( (level >= 14) && (power) ) {
            power--;
            if ( (level == 15) && (power) )
                  power--;
      }
      else {
            dual = 5600 * (BURST_PULSE_COUNT+1);                  // max = 2800
            if ( (differ_values[0] + differ_values[2] > dual) ||
                   (differ_values[1] + differ_values[3] > dual) ) {
                  if (power)
                        power--;
                  dual = 6400 * (BURST_PULSE_COUNT+1);            // over = 3200
                  if ( (differ_values[0] + differ_values[2] > dual) ||
                         (differ_values[1] + differ_values[3] > dual) )
                        if (power)
                              power--;
            }
            else {
                  dual = 1200 * (BURST_PULSE_COUNT+1);            // min =  600
                  if ( (differ_values[0] + differ_values[2] < dual) ||
                         (differ_values[1] + differ_values[3] < dual) )
                        if (power < 15)
                              power++;
            }
      }
      if ( (pow_lvl >> 4) != power ) {
            control_table[5] &= 0x0F;
            control_table[5] |= (power << 4);
            set_tx_power ();
      }
      return;
}
```

☑ Corrected: Register Variables overwritten by switch/case with long

```
#include <reg51.h>

unsigned long dwParam;

unsigned char func(unsigned char bData)  {
  unsigned char bResult;

  switch (dwParam)
  {
    case 0x01234567U:
      bResult = bData;
      break;

    case 0x89ABCDEFU:
      bResult = bData + 1;
      break;

    default:
      bResult = 0;
      break;
  }

  return(bResult);
}
```

☑ Corrected: Fatal Error Cannot Optimize Function in following example

```
#pragma LARGE
```

```
extern bit memcpy (void *s1, void *s2, int n);

bit R (unsigned int ui, unsigned int F, unsigned char *pD, unsigned int D) {
 unsigned char code * p;
 unsigned int L;
 unsigned int N;

 N = ui;
 while (D && N < 98){
   p = 0xE0 + (N * 32);
   if (D <= 32){
     memcpy(pD, p, D);
   } else {
     L = 32 - F;
     memcpy(pD, p, L);
   }
 }
 return (0);
}
```

☑ Corrected: Compiler does not setup R0 for MOVB instruction in MODE2 mode

```
#pragma OPTIMIZE (9,SIZE) BROWSE cd ob pl(30000) mode2 DEBUG OBJECTEXTEND

unsigned char xdata buffer[260];
unsigned long l;

void testfunc (void)
{

      l=(*(unsigned long*)(buffer+7)); // R0 setup missing
      l++;
      for( ;l!=0;l--)
//                          for( l=(*(unsigned long*)(buffer+7));l!=0;l--)
      {
            return ;
      }

}
```

☑ Corrected: Common Subexpr. Problem

```
#define TST_CASTINT_C

typedef unsigned char uchar;
typedef unsigned int uint;

uint uiA;

void main() {
  char   cA = -40;
  uchar ucA = 20;
  int iA;

#if 0
//1. Korrektes Ergebnis durch cast-Operator  (iA = -20)
  iA = (int)cA + ucA;

#endif
//2. Die vorzeichenbehafte Variable wird auf unsigned gecastet
  uiA = (uint)ucA + ((uchar)cA);  // (uiA = 236);

//3. Die vorzeichenbehafte Variable wird auf unsigned int gecastet
```

```
  uiA = (uint) cA + ucA;           // (uiA = 65516)

  while (1);
}
```

☑ Corrected: Compiler generates wrong optimization for CLR C, CJNE construct when OMF2 directive is applied

```
#pragma O2
unsigned char c1, c2;
unsigned char e;

void main (void)  {

  if (c2 == 5)  {
    if (!(c1 == 0x9E))  {
      e = 1;
      goto label_ret;
    }
  }
  else  {
     e = 1;
     goto label_ret;
  }  e = 2;
label_ret: ;
}
```

☑ Corrected: Register variable incorrectly assigned

```
sfr   P6     = 0xFA;
sfr   P7     = 0xDB;
sfr   P8     = 0xDD;

void  SUB_Delay (unsigned int Number);
bit   KEY_KeyPressed;

unsigned int KEY_GetKey (void) {
  unsigned char  key_bd_p1;
  unsigned char  key_bd_p2;
  unsigned char  key_bd_p3;
  unsigned int   key_bd;
  unsigned char  key1, key2, key3;

  key_bd_p1  = 0xFF;
  key1       = 0xFF;
  key_bd_p2  = 0xFF;
  key2       = 0xFF;  // was re-assigned
  key_bd_p3  = 0xFF;
  key3       = 0xFF;

  key_bd_p1 = (P6 | 0x7E);
  key_bd_p2 =  P7;
  key_bd_p3 = (P8 & 0x70) | 0x8F;
  SUB_Delay (5);
  key1      = (P6 & 0x81) | 0x7E;
  key2      =  P7;
  key3      = (P8 & 0x70) | 0x8F;

  if ((key_bd_p1 != key1) | (key_bd_p2 != key2) | (key_bd_p3 != key3)) {
    KEY_KeyPressed = 0;
    return 0;
  }
  return key_bd;
```

```
}
NOTE: Problem was introduced with a new optimizer in C51 V6.14c
CORR: REG51.C(calcwut /13.10.2001/)
```

☑ Corrected: Warning 'single line comment with line continuation' given

---

☑ Enhancement: long cmp 0 generates now more compact code

☑ Corrected: Following Code generates incorrect common blocks due to code rearrange

```
struct s  {
  char a[5];
  char x;
  char y;
};

struct s xdata *p;
struct s xdata *q;
struct s xdata *a;
struct s xdata *b;
char xc;

void func (char x, char y)  {
  p = b;
  if (p->x == x)  return;

  p = b;
  if (b->x == y)  return;

  p = q;
}
```

☑ Corrected: PUSH/POP XPAGE1SFR unbalanced

```
#pragma varbanking (1) OMF251
TestInt (void) interrupt 1 using 2  {
  unsigned char xdata a;
  a += 2;
}
```

☑ Corrected: Internal errors generated

```
typedef char BYTE;
typedef unsigned char  UBYTE;

static code const UBYTE X1[2][0x66];

UBYTE   format = 0;
extern BYTE ReadByte (void);

static BYTE VerifyGMVLX1Config (void)  {
   UBYTE r1L=0;
```

```
   UBYTE r2L=0;
   UBYTE r1R=0;
   UBYTE r2R=0;
   BYTE  rc = 0;

   r1L = ReadByte();
   r2L = ReadByte();
   r1R = ReadByte();
   r2R = ReadByte();

   if ((r1L != X1[((format==0x00)?0:1)][0x0e]) ||
       (r2L != X1[((format==0x00)?0:1)][0x0b]) ||
       (r1R != X1[((format==0x00)?0:1)][0x0e]) ||
       (r2R != X1[((format==0x00)?0:1)][0x0b]) )
     rc = 2;
   return rc;
}
```

☑ Corrected: 'far' mSpace lost in explicite cast

```
#pragma LARGE ROM(D512K) VARBANKING OMF251 DEBUG CODE
#define ADR 0x200000

//#include <Dallas\REG390.H>
//#include <stdio.h>
//#include <stdlib.h>
//#include <string.h>
//#include <absacc.h>

#define FVAR(object, addr)  (*((object volatile far *) ((addr)+0x10000L)))

typedef struct _PAIR  {
  int key;
  int value;
}
PAIR,*PPAIR;

#define BASICOBJECT FVAR(char,ADR)
#define BASICADDRESS FVAR(char *,ADR)
#define COMPOSITEOBJECT FVAR(PAIR,ADR)
#define COMPOSITEADDRESS FVAR(PPAIR,ADR)

char * far char_pointer;
char * far char_pointer1;
unsigned long ptr1;

void test (void)  {
  char_pointer = &COMPOSITEOBJECT;
  char_pointer = (char far *) &COMPOSITEOBJECT;
//  ptr1=(unsigned long)char_pointer;
}
```

☑ Corrected: Dallas 390: DPX1 is not saved on interrupt entry when using dual DPTR

```
#pragma LARGE ROM(D512K) OMF251 MODDP2 DEBUG CODE

extern void x (void);

void test (void) interrupt 1 using 1  {
  x ();
}
```

☑ Corrected: Fails to save ?C?CASE argument in R7 for re-use

```
#pragma OPTIMIZE (9,SIZE) DEBUG OBJECTEXTEND

unsigned char xdata BUF[260];

void main(void) compact  {
  unsigned char data i;

  switch(BUF[4])  {                          // MOV R7,A missing
    case 51:
    case 48:      i=48;  break;
    case 35:
    case 32:      i=32;  break;
    case 19:
    case 16:      i=4;   break;
    case 8:       i=8;   break;
    default:      if (BUF[4]>=96)  i=32;    // re-use of R7
                  else             i=16;
                  break;
  }
}
```

☑ Corrected: Redundant assignments yield to error CANNOT OPTIMIZE FUNCTION

```
void exit (int x)  {
  long a;
  while(1) a = x;
}


void main (void)  {
  float f10, f11, f2, f3;
  double d1, d2;
  f10 = 0; f2 = 1; f3 = 1; d1 = 1; d2 = 1;
  f10 =d1=f2=d2=f3= -8.42e1;
  f11 = 1; f2 = 2; f3 = 2; d1 = 2; d2 = 2;
  f11 =(d1=(f2=(d2=(f3= -8.42e1))));
  if (f10 != f11)
      goto fail;
  exit(0);

fail:
  exit(1);
}
```

☑ Corrected: Optimization fails

```
#pragma OT(9, SIZE)
extern bit cpyx2x (unsigned char xdata *s1,
                   unsigned char xdata *s2,
                   unsigned int);
extern unsigned char xdata T[3];
extern unsigned char xdata P[3];

void v (unsigned char i)   {
  unsigned char xdata a[32];

  switch(i)  {
    case 0:
    case 2:
    case 6:
      cpyx2x(&a[2], T, 3);    // parameter incorrectly set
```

```
      break;
    default:
      cpyx2x(&a[2], P, 3);
  }
}
```

☑ Corrected: MODDA: div16/div16

```
Dallas Datasheet documents wrong sequence for div16/div16.  Problem
is corrected in C51-Library (uidiv.a51  /12.11.2001/)
```

☑ Corrected: DS390: Potential Problem with Float compare

```
#pragma LARGE ROM(D512K) O2 CODE
double dabs  (double d);
double Delta = 0.0;

int dequals  (double val1, double val2)  {
 double *pd;
 pd = &val2;


  if (dabs(val1) == 0.0 && dabs(val2) < Delta) ;
  else if (dabs(val2) == 0.0 && dabs(val1) < Delta) ;
  else if ((dabs(val1 - val2) / dabs(*pd)) > Delta)   return 0;
  return 1;
}
```

☑ Corrected: Optimization fails

```
#pragma LARGE

#define FLTDISPRATE_250MS     0
#define FLTDISPRATE_500MS     1
#define FLTDISPRATE_750MS     2
#define FLTDISPRATE_1S        3
#define FLTDISPRATE_1_5S      4
#define FLTDISPRATE_2S        5
#define FLTDISPRATE_4S        6
#define FLTDISPRATE_MAX       6

main() {
      float dFltDispRate;
      float result;
      unsigned char ucTemp;

      ucTemp=0;
//  = ReadNvm(NVM_FLTDISPRATE);      // Display Rate
    if(ucTemp > FLTDISPRATE_MAX)  {
       ucTemp = FLTDISPRATE_250MS;
//     WriteNvm(NVM_FLTDISPRATE,ucTemp);
    }
    switch (ucTemp)  {
        case FLTDISPRATE_250MS:
            dFltDispRate = 0.25;
            break;
        case FLTDISPRATE_250MS + 20:
            dFltDispRate = 0.25;
            break;
        case FLTDISPRATE_500MS:
            dFltDispRate = 0.5;
            break;
        case FLTDISPRATE_750MS:
```

```
            dFltDispRate = 0.75;
            break;
        case FLTDISPRATE_1S:
            dFltDispRate = 1.0;
            break;
        case FLTDISPRATE_1_5S:
            dFltDispRate = 1.5;
            break;
        case FLTDISPRATE_2S:
            dFltDispRate = 2.0;
            break;
        case FLTDISPRATE_4S:
            dFltDispRate = 4.0;
            break;
        default:
            dFltDispRate = 0.252;
            break;
    };
      result = dFltDispRate;
}
```

☑ Corrected: Following code fails to setup R1/R2/R3 before CLDOPTR

```
extern unsigned char xdata DummyVariable;
extern unsigned char xdata Temp;

void TestFunction(void)  {
  *(&DummyVariable + (Temp * 3)) = 0;
}
```

☑ Corrected: DS390: with Optimize(9) AJMP has wrong encoding in common block

---

☑ Corrected: MX51: following code fails

```
char * const cp1 = (char *) 0x0815;
extern void exit (int);
void main () {
  if (cp1 != (char *)0x815)   exit (5);
  exit (0);
}
```