```
copyright   = Josef Franz Vögel
www         = http://members.aon.at/voegel or  http://www.mcselec.com
comment     = AVR-DOS compiler library
libversion  = 5.5
date        = 13 February 2006
statement   = No SOURCE code from the library may be distributed in any form unless you have permission
statement   = from the author
History     = No known bugs
History     = Version 5.5
History     = - bug in MKDIR fixed (restoring SW-Stack (Y) Pointer)
History     = Version 5.4 (needs BASCOM-AVR 1.11.8.1 or higher)
History     = - Support for Option "Config Clock = User" implemented
History     =   _GetDateTime is called in Date-Time related functions CreateFile, MKDir, Close and
History     =   Flush if USERCLOCK=1 to refresh the date/time variable _sec, _min, _hour, _day, _month,
History     =   _year
History     = - If no CLOCK is implemented (SOFT or USER) default value 2001-01-01 01:00:00 is used
History     =   BASCOM-AVR Version 1.11.8.1 is needed for this feature
History     = - New Parameter cFATSecondUpdate to increase Write speed to the card. Suppressing the
History     =   updating of the FAT Copy on the drive reduces the worst case delay time of AVR-DOS if
History     =   a FAT-Sector is full (f.E.: after 500kB of a 128MB Card/FAT16) and a new FAT-Sector has
History     =   to be loaded.
History     = - New Parameter cVariableSeparator: Normally a Comma is used to separate variables
History     =   written with WRITE and read with INPUT. This new parameter allow other characters like
History     =   TAB for EXCEL TAB separated files
History     = Version 5.3
History     = - Bug fixed in _GetFileSystem, FAT Type was sometimes wrong reported
History     =   if T-Flag was changed in mass-Storage Driver (CF,HD,SD,MMC) at reading
History     =   the PBR sector
History     = Version 5.2
History     = - Bug fixed in the NAME (renaming a file) routine: _FileReName
History     = Version 5.1
History     = - Added Create Date/Time to directory entry of file name and Sub-Directories
History     = - read and write of Double variable in ASCII and Binary mode
History     = Version 5.0
History     = - Update with directory and FAT32 support
History     = Version 4.2
History     = - Added FAT16 Type with signature 0E
History     = Version 4.1
History     = - fixed bug in _SaveFileSector routine, which corrupted at Close/Flush
History     =   Byte 504 - 507 in FAT-Sector
History     = Version 4
History     = - added parameter for "lazy" saving FAT and Directory information, please check
History     =   configuration of Config_AVR-DOS.BAS. It reduced write frequency of FAT and Directory
History     =   Sector, and increases writing speed of data
History     = Version 3
History     = - fixed bug in _DriveReadSector_SEV and _DriveWriteSector_SEV
History     = Version 2:
History     = - added parameter 0 for _GetFileSystem is ignoring MBR and loads PBR at sector 0 for
History     =   disks without a master boot record (MBR)
History     = - added Check for valid Sectors per Cluster in _GetFileSystem
History     = - bug fixed for Reading in binary mode ; file-pointer was destroyed at loading new sector


   ; Routines for AVR-DOS
[_AVRDOSVER]
   ;returns lib version of AVR-DOS
_AVRDOSVer:
  ldi r16,55                            ; return version
  Ret
[END]
```

```
[_ClearFileSystem]
_ClearFileSystem:
   ;---------------------------------------------------------------------------
   ; Clear the SRAM used by AVR-DOS
   ;---------------------------------------------------------------------------
   loadadr gbDOSError , Z                ; clear area of File -system in SRAM
 * ldi r26 , lbyte(c_FileSystemSRAMSize)
 * ldi r27 , hbyte(c_FileSystemSRAMSize)
   clr r25
_ClearFileSystem1:
   st Z+, r25
   sbiw r26, 1
   brne _ClearFileSystem1
   ret
[end]


[_CheckFileSystemType]
$EXTERNAL _SetDOSError
   ;---------------------------------------------------------------------------
   ; Check file system type in r24 passend
   ; C-Flag set and errocode in r25, if invalid file system
   ; T-Flag set if FAT32 File System
   ;---------------------------------------------------------------------------
_CheckFileSystemType:
   clt                                   ; FAT16 Flag
   cpi r24, &H04                         ; FAT16 <= 32 MB
   breq _CheckFileSystemType8
   cpi r24, &H06                         ; FAT16 >= 32MB
   breq _CheckFileSystemType8
   cpi r24, $H0E                         ; FAT16 with Bios Extension
   breq _CheckFileSystemType8
   set                                   ; FAT32 Flag
   cpi r24, &H0B                         ; FAT32 End of partition below 8GN
   breq _CheckFileSystemType8
   cpi r24, &H0C                         ; FAT32 End of partition over 8GB
   breq _CheckFileSystemType8
 * ldi r25 , cpFileSystemNotSupported    ; no supported file system found
   rjmp _SetDOSError
_CheckFileSystemType8:
   lds r0, {gbFileSystemStatus}          ; store FAT16/32 Flag to gbFileSystemStatus
 * bld r0, dFileSystemStatusFAT
   sts {gbFileSystemStatus}, r0
   clc
   ret
[end]


[_CheckBootSign]
$EXTERNAL _SetDOSError , _WordZ2B
   ;---------------------------------------------------------------------------
   ; Check boot sign in Master boot record (MBR) and partition boot record (PBR)
   ; for 55AA at byte 511/512
   ; used from _GetFileSystem
   ; Input-setting: Errorcode in r25
   ;---------------------------------------------------------------------------
_CheckBootSign:
   loadadr gbDirBuffer(511) , Z
   rcall _WordZ2B
   subi r20, &H55
   sbci r21, &HAA
   brne _CheckBootSign1
   ret
_CheckBootSign1:
   rjmp _SetDOSError
[end]
```

```
[_GetFileSystem]
$EXTERNAL _A2LongX , _Add32_Word , _B2LongZ , _B2WordX , _CheckBootSign , _CheckFileSystemType
$EXTERNAL _ClearDOSError , _ClearFileSystem , _Cluster2Sector , _LongX2A , _LongX2B , _LongZ2A
$EXTERNAL _Mem_Copy , _Mov4A2B , _ReadDirSector , _Sector2Cluster , _SetDOSError , _ShiftR_WordB
$EXTERNAL _WordX2B , _WordZ2A , _WordZ2B
$EXTERNAL _SUB32
   ;-------------------------------------------------------------------------------
   ; Load File System Area with values from File-System on Disk
   ; check for validy of the Disk-file-system according to AVR-DOS
   ; Input: Partition Number in r24, 1-based
   ;-------------------------------------------------------------------------------
_GetFileSystem:
   st -Y, r24                           ; save partiton#
   rcall _ClearFileSystem
   ; use Dir-Handling area to load MBR, glDirSectorNumber was cleared too, so load sector 0
   ; will be loaded
   rcall _ReadDirSector
   brcs _GetFileSystemErrorExitY1
   ; Check MBR Sign (55AA)
 * ldi r25 , cpNoMBR
   rcall _CheckBootSign
   brcs _GetFileSystemErrorExitY1
   ld r24, Y                            ; partition#
   tst r24                              ; handle partition# 0 same as 1
   breq _GetFileSystem5a
   loadAdr gbDirBuffer(447) , Z         ; Z points to first partiton Status
_GetFileSystem1:
   dec r24
   breq _GetFileSystem2
   adiw zl, 16                          ; offset for one partition table
   rjmp _GetFileSystem1
_GetFileSystem2:
   ld r24, Z                            ; check partition status byte
   cbr r24, &H80                        ; should be H00 or H80
   brne _GetFileSystem3
   adiw zl, 4                           ; skip to file system type
   ld r24, Z                            ; load file system type byte
   rcall _CheckFileSystemType
   brcc _GetFileSystem4
_GetFileSystem3:
   ld r24, Y                            ; restore Y-Stack and get partition#
   cpi r24, 2                           ; only check for valid boot partition
   brcs _GetFileSystem5a
 * ldi r25, cpFileSystemNotSupported
   rjmp _SetDOSError
_GetFileSystemErrorExitY1:
   adiw yl, 1
   rjmp _SetDOSError
_GetFileSystem4:
   sts {gbFileSystem}, r24
   adiw zl, 4                           ; skip to partition first sector#
   loadadr glDirSectorNumber , X        ; Sector -Number for desired partition
   rcall _Mem_Copy_4                    ; Copy Partition First Sector (4 Bytes) to SRAM
   ; Now load Partition Boot Record
_GetFileSystem5:
   rcall _ReadDirSector
   brcs _GetFileSystem3                 ; set DOS-Error once more, but saves code
   ; Check PBR Sign (55AA)
 * ldi r25 , cpNoPBR
   rcall _CheckBootSign
   brcs _GetFileSystemErrorExitY1
   ; get File system paramters from Partition Boot Record
_GetFileSystem5a:
   ld r24, Y+                           ; restore Y-stack
   ; check bytes per sector: only 512 bytes per sectors supported
   Loadadr gbDirBuffer(12) , Z          ; Bytes per Sector
   rcall _WordZ2B
   subi r20, 0                          ; 512
   sbci r21, 2
 * ldi r25 , cpSectorSizeNotSupported
   brne _GetFileSystemErrorExit
   ; calculate FATFirstSector = PartitionFirstSector + Reserved Sectors
   loadadr gbDirBuffer(15) , X          ; reserved Sectors is offset of 1.FAT from Partition start
   rcall _WordX2B
```

```
   loadAdr glDirSectorNumber , X
   rcall _LongX2A                       ; Sector number of partition-start to registers
   rcall _Add32_Word                    ; calculate FATFirstSector
   loadadr glFATFirstSector , X
   rcall _A2LongX

   ; Sectors per FAT
   loadadr gbDirBuffer(23) , X          ; word for FAT16
   rcall _WordX2B_L
   subi r20, 0                          ; check for zero
   sbci r21, 0
   brne _GetFileSystem7
_GetFileSystem6:
   loadadr gbDirBuffer(37) , X          ; Long for FAT32
   rcall _LongX2B
_GetFileSystem7:
   loadadr glSectorsPerFAT , Z          ; store it
   rcall _B2LongZ

   ; Load number of FATs
   lds r24, {gbDirBuffer+16}            ; Number of FATs is at offset 16 in Boot Sector
   sts {gbNumberOfFATs}, r24            ; store it
 * ldi r25 , cpCountOfClustersNotSupported
   tst r24                              ; test for 0
   breq _GetFileSystemErrorExit
   cpi r24, 3                           ; test for >= 3
   brcs _GetFileSystem7a
_GetFileSystemErrorExit:
   rjmp _SetDOSError
_GetFileSystem7a:
   ; calculate RootfirstSector on FAT16 (and DataFirstSector on FAT32)
   rcall _Add32_r                       ; same as _Add32 but relative call
   sbrc r24, 1
   rcall _Add32_r                       ; add once more, if two FATs
   loadadr glRootFirstSector , X
   rcall _A2LongX

   ; Root Entries
   loadadr gbDirBuffer(18) , X
   rcall _WordX2B                       ; load from Boot sector
   loadadr gwRootEntries , X
   rcall _B2WordX                       ; store it, on FAT32 it should be 0
   ldi r24, 4
   rcall _ShiftR_WordB                  ; RootEntries / 16 = number of Root Sectors
   rcall _Add32_Word                    ; calculate first Data Sector
   loadadr glDataFirstSector , X
   rcall _A2LongX

_GetFileSystem8:
   ; calculate used sectors for FAT and Root (on FAT16)
   loadadr glDirSectorNumber , X
   rcall _LongX2B
   call _Sub32
   rcall _Mov4A2B

   ; load partition size
   loadadr gbDirBuffer(20) , Z          ; Partition Size small
   rcall _WordZ2A_L
   subi r16, 0                          ; check to zero
   sbci r17, 0
   brne _GetFileSystem9
   loadadr gbDirBuffer(33) , Z          ; partition size large
   rcall _LongZ2A
_GetFileSystem9:
   ; calculate Data size of partition
   Call _Sub32                          ; A: count of sectors in data area
   ; Sectors per Cluster and check for valid entry
   lds r24, {gbDirBuffer+13}
   sts {gbSectorsPerCluster}, r24
 * ldi r25 , cpSectorsPerClusterNotSupported        ; SectorsPerCluster have an error
   tst r24                              ; only 1, 2, 4, 8, 16, 32 or 64 allowed
   breq _GetFileSystemErrorExit         ; that is only one bit in the byte from bit0 to bit6
_GetFileSystem11:
   lsr r24                              ; check for only one bit set
```

```
   brcc _GetFileSystem11
   brne _GetFileSystemErrorExit
   rcall _Sector2Cluster
   rcall _Add32_1                       ; Cluster number begins with 2
   loadadr glMaxClusterNumber , X
   rcall _A2LongX
   ; Check, whether FAT Type is determined by partition type byte
   lds r0, {gbFileSystemStatus}         ; store FAT16/32 Flag to gbFileSystemStatus
 * bst r0, dFileSystemStatusFAT
   lds r24, {gbFileSystem}
   tst r24
   brne _GetFileSystem11a
   ; File type unknow, check by maxClusterNumber
   ; Check for FAT-Type (MaxClusterNumber >= 65526 (H0000FFF6) is FAT32)
   clt                                  ; flag for FAT16
   ldi r24, &H06                        ; default FAT16 Type
   subi r16, &HFF
   sbci r17, &HF6
   sbci r18, &H00
   sbci r19, &H00
   brcs _GetFileSystem11b
   set                                  ; flag for FAT32
   ldi r24, &H8B                        ; default FAT32 type
_GetFileSystem11b:
   sts {gbFileSystem}, r24              ; store default File System type
   clr r25
 * bld r25, dFileSystemStatusFAT        ; save FAT-Type flag
   sts {gbFileSystemStatus}, r25
_GetFileSystem11a:
   brtc _GetFileSystem12
   ; Type is FAT32; get cluster of root directory and calculate first sector
   loadadr gbDirBuffer(45) , X
   rcall _LongX2A
   rcall _Cluster2Sector
   loadadr glRootFirstSector , X
   rcall _A2LongX
_GetFileSystem12:
   ldi r24, 2                           ; first cluster# to search is 2, all ather bytes were
   sts {glLastSearchedCluster}, r24     ; initialised with 00 at start
   loadadr glRootFirstSector , X        ; store RootFirstSector to
   rcall _LongX2A
   loadadr glDirFirstSectorNumber , X   ; DirFirstSectorNumber
   rcall _A2LongX
   loadadr glFreeDirSectorNumber , X    ; FreeDirSectorNumber
   rcall _A2LongX
   rjmp _ClearDOSError
[end]
```

```
[_SaveA2LongY]
   ;---------------------------------------------------------------------------
   ; Load r16-19 to Softstack
   ;---------------------------------------------------------------------------
_SaveA2LongY:
   sbiw yl, 4
_A2LongY:
   st Y+0, r16                          ; save number of free cluster
   st Y+1, r17
   st Y+2, r18
   st Y+3, r19
   ret
[end]


[_LongY2A]
   ;---------------------------------------------------------------------------
   ; load 4 Bytes from Softstack (Y+0 .. Y+3) to R16-19
   ;---------------------------------------------------------------------------
_LongY2A:
   ld r16, Y+0
   ld r17, Y+1
   ld r18, Y+2
   ld r19, Y+3
   ret
[end]


[_InitDirEntry]
$EXTERNAL _ClearDirEntry , _Mem_Copy , _SetDirFATWritePending , _SetDirTimeDate
   ;---------------------------------------------------------------------------
   ; initialize a Dir-Entry with Filename, Attribut, First-Cluster, Date, Time
   ; Z: DirEntry
   ; gsTempFileName: FileName
   ; A: First Cluster#
   ; r24: File-attribut
   ;---------------------------------------------------------------------------
_InitDirEntrySub:
   ldi r24, &H10                        ; directory attribut
_InitDirEntry:
   push zh
   push zl
   ori r24, &H20                        ; set archiv bit
   push r24
   rcall _ClearDirEntry
   pop r24
   st Z+11, r24                         ; store file-attribut
   st Z+26, r16                         ; low Word of cluster#
   st Z+27, r17
   st Z+20, r18                         ; on FAT16 r18 and r19 is 00, so no
   st Z+21, r19                         ; problem to store whithout checking FAT 16/32
   mov xl, zl
   mov xh, zh
   loadadr gsTempFileName , Z
   ldi r24, 11
   rcall _Mem_copy
   ;   adiw xl, 11
   adiw xl, 3                           ; goto Creation Date / time
   call _SetDirTimeDate                 ; write creation date / time
   adiw xl, 4                           ; goto write date / time
   call _SetDirTimeDate                 ; write  date and time
   rcall _SetDirWritePending
   pop zl
   pop zh
   ret
[end]
```

```
[_MKDir]
$EXTERNAL _Check_Root , _ClearA , _ClearFileSector , _GetFreeCluster , _GetFreeDirEntry
$EXTERNAL _InitDirEntry , _LastCluster2FAT , _LongY2A , _NormFileName , _SaveA2LongY , _SaveSector_DIR
$EXTERNAL _SaveSector_FAT , _SearchDirEntry , _Sector2Cluster , _SetDOSError , _SetFileNameError
   ;--------------------------------------------------------------------------------
   ; create a new Directory
   ; Input X: name of new directory
   ;--------------------------------------------------------------------------------
_MkDir:
   rcall _NormFileName                  ; filename now at Z-Pointer
   breq _MkDir1
   rjmp _SetFileNameWildCardError0      ; Set DOSError for Wildcard in Filename
_MkDir1:
   rcall _SearchDirEntry                ; Pathname exists ?
   brcs _MkDir1a
 * ldi r25 , cpFileExists
   rjmp _SetDOSError
_MkDir1a:
   rcall _GetFreeCluster
   brcs _MkDir7
   rcall _LastCluster2FAT               ; no free cluster available
   rcall _SaveA2LongY                   ; put A to Stack (y-4)
   rcall _GetFreeDirEntry
   brcs _MkDir6                         ; no free DirEntry found
   rcall _LongY2A                       ; cluster# fack from Y-Stack
   rcall _InitDirEntrySub
   rcall _LongY2A                       ; free cluster#
   call _ClearDirCluster                ; clear all sectors of this cluster an set Pointer to first
                                         sector
   ; make dot entry for directory
   ; Prepare file name buffer with ".          "
   loadadr gsTempFileName , X
   ldi r24, asc(".")
   st X+, r24
   ldi r24, &H20                        ; append 10 blank
   ldi r25, 10
_MkDir2:
   st X+, r24
   dec r25
   brne _MkDir2
   rcall _LongY2A                       ; first cluster of directory
   rcall _InitDirEntrySub
   push zh
   push zl
   ; make dot dot entry for directory
   rcall _Check_Root
   breq _MkDir2a
   rcall _Sector2Cluster
   rjmp _MkDir2c
_MkDir2a:
   rcall _ClearA                        ; first cluster# is 0000 on Root
_MKDir2c:
   ldi r24, asc(".")
   sts {gsTempFileName+1}, r24          ; second dot into file name buffer
   pop zl
   pop zh
   adiw zl, 32                          ; skip to second dir-entry
   rcall _InitDirEntrySub
*#IF cSepFATHandle = 1
   call _SaveSector_FAT
   brcs _MkDir6
   call _SaveSector_Dir
*#ELSE
   call _SaveSector_FAT
*#ENDIF
_MkDir6:
 * in r24, SREG
   adiw yl, 4
 * out SREG , r24
_MkDir7:
   ret

   ;§§§§§§§ Im Fehlerfalle EOC für SUB Dir wieder löschen
[end]
```

```
[_Check_Root]
$EXTERNAL _LongX2A , _LongX2B
$EXTERNAL _COMP32
   ;---------------------------------------------------------------------------
   ; Checks whether current Directory is Root Directory
   ; Compare current First Dir Sector with RootFirstSector
   ; Z-Flag is set, if true
   ;---------------------------------------------------------------------------
_Check_Root:
   loadadr glDirFirstSectorNumber , X
   rcall _LongX2A
   loadadr glRootFirstSector , X
   rcall _LongX2B
   jmp _Comp32
[end]
```

```
[_ChDir]
$EXTERNAL _A2LongX , _A2WordX , _CheckA_Zero , _ClearA , _ClearDOSError , _Cluster2Sector , _GetSector
$EXTERNAL _LongX2A , _NormFileName , _SearchDirEntry , _SetDOSError , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; change to a new Directory
   ; Input: "Pathname" -> change to sub-directory patname
   ;         "." -> change tp upper directory
   ;         "\" -> change to root directory
   ; Input X: name of directory to change
   ;---------------------------------------------------------------------------
_ChDir:
   ld r24, X
   cpi r24, ASC(".")
   brne _ChDir4
_ChDir0:
   loadadr glDirFirstSectorNumber , X
   rcall _LongX2A
   rcall _GetSector_Dir
   adiw zl, 32                          ; skip to ".." Entry
   ld r24, Z
   cpi r24, ASC(".")
   brne _ChDir61                        ; ".." Entry not found
   lds r0, {gbFileSystemStatus}
 * bst r0, dFileSystemStatusFAT
   clr r18
   clr r19
   ld r16, Z+26
   ld r17, Z+27
   brtc _ChDir1
   ld r18, Z+20
   ld r19, Z+21
_ChDir1:
   rcall _CheckA_Zero
   brne _ChDir3
_ChDir2:
   loadadr glRootFirstSector , X
   rjmp _ChDir8
_ChDir3:
   rcall _Cluster2Sector
   rjmp _ChDir9
_ChDir4:
   cpi r24, ASC("\")
   breq _ChDir2
_ChDir5:
   rcall _NormFileName                  ; filename now at Z-Pointer
   breq _ChDir6
   rjmp _SetFileNameWildCardError0      ; Set DOSError for Wildcard in Filename
_ChDir6:
   rcall _SearchDirEntry                ; Pathname exists ?
   brcs _ChDir60
   ld r24, Z+11                         ; Get Attribut of Dir - Entry
```

```
   cbr r24, &B00110000                  ; check for only Directory and Archiv-Bit Set
   brne _ChDir62                        ; not a Sub - Directory Entry?   Fehlercode $$$$$
 * lds r0 , {gbFileSystemStatus}
   clr r18
   clr r19
   ld r16, Z+26
   ld r17, Z+27
 * sbrs r0 , dFileSystemStatusFAT
   rjmp _ChDir7
   ld r18, Z+20
   ld r19, Z+21
_ChDir7:
   rcall _Cluster2Sector
   rcall _GetSector_Dir
   brcs _ChDir60
   loadadr glDirSectorNumber , X
_ChDir8:
   rcall _LongX2A
_ChDir9:
   loadadr glDirFirstSectorNumber , X
   rcall _A2LongX
   loadadr glFreeDirSectorNumber , X
   rcall _A2LongX
   loadadr gwFreeDirEntry , X
   rcall _ClearA
   rcall _A2WordX
   rjmp _ClearDOSError
_ChDir60:
   ret
_ChDir61:
 * ldi r25, cpSubDirectoryError
   rjmp _SetDOSError
_ChDir62:
 * ldi r25, cpNotASubDirectory
   rjmp _SetDOSError
[End]


[_RMDir]
   ;-----------------------------------------------------------------------------
   ; Remove a sub-directory in current directory
   ; Input X: -> Directory name
   ; checks whether sub-directory is empty
   ;-----------------------------------------------------------------------------
$EXTERNAL _ChDir , _CleanHWStack , _DeleteFile , _NormFileName , _SearchDirEntry , _SetDOSError
_RMDir:
   push xh                              ; save Pointer to Dir-name
   push xl
   rcall _ChDir5                        ; Change to SubDir without considering . and \
   brcs _RMDir4                         ; SubDir not exists?
   loadadr gsTempFileName , X
   ldi r24, asc("?")
   ldi r25, 11                          ; 11 ? is wildcard search
_RmDir2:
   st X+, r24
   dec r25
   brne _RMDir2
   loadadr gsTempFileName , Z
   rcall _SearchDirEntry
   push r25                             ; save error code: file Found?
   rcall _ChDir0                        ; goto parent directory
   brcs _RMDir5
   pop r25
   tst r25
   breq _RMDir6
 * cpi r25, cpFileNotFound
   brne _RMDir4
   pop xl
   pop xh
   rcall _NormFileName
   rcall _SearchDirEntry
   brcs _RMDir9
   rjmp _DeleteFile8
```

```
_RMDir5:
   rjmp _CleanHWStack3
_RmDir6:
 * ldi r25, cpSubDirectoryNotEmpty
   rcall _SetDOSError
_RMDir4:
   rjmp _CleanHWStack2
_RMDir9:
   ret
[end]


[_FileRename]
$EXTERNAL _ClearDOSError , _Mem_Copy , _NormFileName , _SaveSector_DIR , _SearchDirEntry
$EXTERNAL _SetDirFATWritePending , _SetDOSError , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; Rename a file or a sub-directory
   ; X: Pointer to new name
   ; Z: Pointer to old name
   ;---------------------------------------------------------------------------
_FileRename:
   st -Y, zh
   st -Y, zl
   st -Y, xh
   st -Y, xl
   rcall _Normfilename
   brne _FileReName5
   rcall _SearchDirEntry                ; does new name exists
 * ldi r25, cpFileExists
   brcc _FileReName6
   ld xl, Y+2                           ; check of existing old file name
   ld xh, Y+3
   rcall _NormFileName
   brne _FileReName5
   rcall _SearchDirEntry
   brcs _FileReName6
   push zh                              ; save dir entry pointer
   push zl
   ld xl, Y+0
   ld xh, Y+1
   rcall _NormFileName
   pop xl                               ; dir entry pointer
   pop xh
   ldi r24, 11
   rcall _Mem_Copy
   rcall _SetDirWritePending            ; bug fixed in 5.2
   adiw yl, 4
   rcall _SaveSector_DIR
   rjmp _ClearDOSError
_FileReName5:
   rcall _SetFileNameWildCardError0
_FileReName6:
   adiw yl, 4
_FileReName7:
   rjmp _SetDOSError
[end]


   ; File Opening Routines called with File Name
   ;
[_FileOpen_Input]
$EXTERNAL _ClearDOSError , _GetFreeFileNumber , _NormFileName , _OpenFile , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; Open a file for Input (Reading) and load first sector
   ; Input:  X-> Filename with 12 char.(Short DOS name 8.3 f.e "Test1234.txt")
   ;         r24: File# or _FileOpen_Input0 without file#
   ; Output: r25: Error-code
   ;         r24: File# for further use if called without file# at _FileOpen_Input0
   ;---------------------------------------------------------------------------
_FileOpen_Input0:
   rcall _GetFreeFileNumber             ; to get free file# in r24
   ; _GetFreeFileNumber do not change X
```

```
   brcs _FileOpen_Input3                ; Error?; if C-set
_FileOpen_Input:
   rcall _NormFileName                  ; Result: Z-> Normalized name
   brne _FileOpen_Input2                ; Wild Card in Filename?
   push r24                             ; save File#
 * ldi r25 , cpFileOpenInput            ; Read mode
   rcall _OpenFile                      ; Search file and set File-handle and load first sector
   pop r24                              ; file#
   brcs _FileOpen_Input3                ; Error at file open ?
   rjmp _ClearDOSError                  ; File opending was OK!
_FileOpen_Input2:
   rjmp _SetFileNameWildCardError0      ; Set DOSError for Wildcard in Filename
_FileOpen_Input3:
   clr r24                              ; Error clears file#
   ret                                  ; Error-code yet loaded in r25
[End]


[_FileOpen_Append]
$EXTERNAL _ClearDOSError , _FileOpen_FileNotExist , _GetFreeFileNumber , _GotoFilePosition
$EXTERNAL _LoadFileSizeYZ2A , _NormFileName , _OpenFile , _SetFileNameError
   ;--------------------------------------------------------------------------------
   ; Open a file for Append (Writing at end of file)
   ; Input:  X-> Filename with 12 char.(Short DOS name 8.3 f.e "Test1.txt")
   ;         r24: File# or _FileOpen_Append0 without file#
   ; Output: r25: Error-code
   ;         r24: File# for further use if called without file# at _FileOpen_Append0
   ;--------------------------------------------------------------------------------
_FileOpen_Append0:
   rcall _GetFreeFileNumber             ; to get free file# in r24
   brcs _FileOpen_Append4               ; Error?; if C-set
_FileOpen_Append:
   rcall _NormFileName                  ; Result: Z-> Normalized name
   brne _FileOpen_Append5               ; Wildcard in name?
   push r24                             ; save File#
 * ldi r25 , cpFileOpenAppend           ; Append mode write at end of file
   rcall _OpenFile                      ; Search file and set File-handle and load first sector
   pop r24                              ; file#
   brcc _FileOpen_Append1               ; File exists?
 * cpi r25 , cpFileNotFound
   brne _FileOpen_Append3               ; other Error than File not found?
 * ldi r25 , cpFileOpenAppend
   rjmp _FileOpen_FileNotExist
_FileOpen_Append1:
   st -Y, zh                            ; file open exits with filehandler in Z
   st -Y, zl
 * ldd r25 , Z + co_FileBufferStatus    ; now positioning , if file empty
 * sbrc r25 , dFileEmpty                ; check file empty(no cluster allocated)
   rjmp _FileOpen_Append01
   rcall _LoadFileSizeYZ2A              ; Go to EOF, load size to A
   push r24                             ; save file#
   clz                                  ; otherwise it it handled as empty file
   call _GotoFilePosition               ; go 1 Byte behind last byte
   pop r24
   brcs _FileOpen_Append2
_FileOpen_Append01:
   rjmp _ClearDOSError_add2toY
_FileOpen_Append2:
   adiw yl, 2                           ; error at file opending, Clear Y-Stack
_FileOpen_Append3:
   clr r24                              ; error set file# to 0
   sec
_FileOpen_Append4:
   ret
_FileOpen_Append5:
   rjmp _SetFileNameWildCardError0      ; Set DOSError for wildcard in name
[End]
```

```
[_FileOpen_Output]
$EXTERNAL _ClearDOSError , _CreateFile , _GetFreeFileNumber , _NormFileName , _SetFileNameError
   ;--------------------------------------------------------------------------
   ; Open a file for OUTPUT (Writing); delete an existing file
   ; Input:  X-> Filename with 12 char.(Short DOS name 8.3 f.e "Test1.txt")
   ;         r24: File# or _FileOpen_Output0 without file#
   ; Output: r25: Error-code
   ;         r24: File# for further use if called without file# at _FileOpen_Output0
   ;--------------------------------------------------------------------------
_FileOpen_Output0:
   rcall _GetFreeFileNumber              ; to get free file# in r24
   brcs _FileOpen_Output3                ; no free file# ??
_FileOpen_Output:
   rcall _NormFileName                   ; prepare filename and check for wildcard
   brne _FileOpen_Output2                ; wildcard in name?
   push r24                              ; save file#
 * ldi r25 , cpFileOpenOutput            ; file open mode
   rcall _CreateFile_DeleteOld           ; create file and delete old if exists
   pop r24                               ; get file#
   brcs _FileOpen_Output3                ; problem at creation?
   rjmp _ClearDOSError                   ; File Creating ended OK!
_FileOpen_Output2:
   rjmp _SetFileNameWildCardError0       ; Set DOSError for Wildcard in filename
_FileOpen_Output3:
   clr r24                               ; Error clears file#
   ret
[End]


[_FileOpen_Binary]
$EXTERNAL _FileOpen_FileNotExist , _GetFreeFileNumber , _NormFileName , _OpenFile , _SetFileNameError
   ;--------------------------------------------------------------------------
   ; Open a file for BINARY (Rwading&Writing); Create it, if it does not exists
   ; Input:  X-> Filename with 12 char.(Short DOS name 8.3 f.e "Test1.txt")
   ;         r24: File# or _FileOpen_Binary0 without file#
   ; Output: r25: Error-code
   ;         r24: File# for further use if called without file# at _FileOpen_Binary0
   ;--------------------------------------------------------------------------
_FileOpen_Binary0:
   rcall _GetFreeFileNumber              ; to get free file# in r24
   ; dont change X
   brcs _FileOpen_Binary3                ; Error?; if C-set
_FileOpen_Binary:
   rcall _NormFileName                   ; Result: Z-> Normalized name
   brne _FileOpen_Binary1                ; Wildcard in file name?
   push r24                              ; File#
 * ldi r25 , cpFileOpenBinary            ; Binary mode
   rcall _OpenFile                       ; Search file and set File-handle and load first sector
   pop r24                               ; file#
   brcc _FileOpen_Binary4                ; Existing file found?
 * cpi r25 , cpFileNotFound              ; check kind of last error
   brne _FileOpen_Binary2
 * ldi r25 , cpFileOpenBinary            ; Binary mode
   rjmp _FileOpen_FileNotExist           ; Create new file
_FileOpen_Binary1:
   rjmp _SetFileNameWildCardError0       ; set DOSError for Wildcard in name
_FileOpen_Binary2:
   sec                                   ; re-set C (Error) flag
_FileOpen_Binary3:
   clr r24                               ; carry not affected
_FileOpen_Binary4:
   ret                                   ; Error-code in r25
[End]


[_FileOpen_FileNotExist]
$EXTERNAL _ClearDOSError , _CreateFile
   ;--------------------------------------------------------------------------
   ; Creates a new file, in case an open command do not find an existing file
   ; used from Append, Binary opening of a file
   ; Input:
   ;       Z-> FileName
```

```
   ;       r24: file#
   ;       r25: file open mode
   ;---------------------------------------------------------------------------
_FileOpen_FileNotExist:
   push r24                            ; save file#
   rcall _CreateFile_CheckedNotExists
   pop r24                             ; get file#
   brcs _FileOpen_FileNotExist1
   rjmp _ClearDOSError                 ; Creating was OK
_FileOpen_FileNotExist1:
   clr r24                             ; Error clears file# too
   ret
[End]


[_FileLoc]
$EXTERNAL _ClearA , _ClearDOSError , _GetFileHandle , _Mem_Copy
   ;---------------------------------------------------------------------------
   ; Location of last read or written byte in file (result is 1-based)
   ; r24 File#
   ; X Pointer to Long-Variable for result
   ;---------------------------------------------------------------------------
_FileLoc:
   rcall _GetFileHandle
   brcc _FileLoc1
   rjmp _ClearA                        ; at Error give 0 back
_FileLoc1:
 * adiw zl , Co_FilePosition           ; get file position
   rcall _Mem_Copy_4
   rjmp _ClearDOSError                 ; = 1-based Location
[End]


[_FileLOF]
$EXTERNAL _ClearA , _ClearDOSError , _GetFileHandle , _Mem_Copy
   ;---------------------------------------------------------------------------
   ; Length of File
   ; r24 File#
   ; X Pointer to Long-Variable for result
   ;---------------------------------------------------------------------------
_FileLOF:
   rcall _GetFileHandle
   brcc _FileLOF1
   rjmp _ClearA                        ; at error give 0 back
_FileLOF1:
 * adiw zl , co_FileSize               ; load filesize to A
   rcall _mem_Copy_4
   rjmp _ClearDOSError
[End]


[_FileSeek]
$EXTERNAL _A2LongX , _Add32_Word , _ClearA , _ClearDOSError , _GetFileHandle , _LongZ2A
   ;---------------------------------------------------------------------------
   ; Location of last read/written Byte; 1-based
   ; r24 File#
   ; X Pointer to Long-Variable for result
   ;---------------------------------------------------------------------------
_FileSeek:
   rcall _GetFileHandle
   brcc _FileSeek1
   rjmp _ClearA                        ; at error give 0 back
_FileSeek1:
 * adiw zl , co_Fileposition           ; file position to A
   rcall _LongZ2A
   rcall _add32_1                      ; 0-based to 1-based
   rcall _A2LongX                      ; and to SRAM result variable
   rjmp _ClearDOSError
[End]
```

```
[_FileSeekSet]
$EXTERNAL _CheckFileAccess_Mode , _ClearDOSError , _GetFileHandle , _GotoFilePosition , _LongX2A
$EXTERNAL _SetErrorFlag
   ;--------------------------------------------------------------------------------
   ; change file position (read/write pointer)
   ; r24 File#
   ; X Pointer to Long-Variable of new position to set
   ;--------------------------------------------------------------------------------
_FileSeekSet:
   rcall _GetFileHandle
   brcs _FileSeekSet2
 * ldi r25 , cFileSeekSet_Mode           ; check allowed for given file open mode
   rcall _CheckFileAccess_Mode_Z
   brcs _FileSeekSet2                    ; not allowed?
   st -Y, zh                             ; File handle address to Y-Stack
   st -Y, zl
   rcall _LongX2A                        ; new position to A
   rcall _GotoFilePosition_1base         ; goto new position
   brcs _FileSeekSet1                    ; Error ?
   rjmp _ClearDOSError_add2toY           ; OK!
_FileSeekSet1:
   rjmp _SetErrorFlag_add2toY            ; Error
_FileSeekSet2:
   ret                                   ; DOSError and C-Flag yet set
[End]


[_FileLineOutput]
$EXTERNAL _ClearDOSError , _FileWriteCRLF , _FileWriteString , _GetFileHandle , _SetErrorFlag
   ;--------------------------------------------------------------------------------
   ; write one line to the file (string + CR + LF)
   ; filen# in r24
   ; String pointer in X
   ;--------------------------------------------------------------------------------
_FileLineOutput:
   rcall _GetFileHandle
   brcs _FileLineOutput2                 ; Error? file# not found?
   st -Y, zh                             ; file handle address to Y0/1
   st -Y, zl                             ;
   call _FileWriteString                 ; write String
   brcs _FileLineOutput1
   call _FileWriteCRLF                   ; write CRLF
   brcs _FileLineOutput1
   rjmp _ClearDOSError_Add2toY           ; OK!
_FileLineOutput1:
   rjmp _SetErrorFlag_add2toY            ; Error; restore Y-pointer from file handle
_FileLineOutput2:
   ret
[End]


[_FileLineInput]
$EXTERNAL _FileReadByte , _GetFileHandle
   ;--------------------------------------------------------------------------------
   ; Read a Line from a File
   ; filenumber in r24
   ; String pointer in X
   ; max String length in r25
   ;--------------------------------------------------------------------------------
_FileLineInput:
   push r25                              ; save string-length
   rcall _GetFileHandle
   pop r25                               ; get string-length back
   brcs _FileLineInput5                  ; Error?
   st -Y, zh                             ; file handle to Y0/1
   st -Y, zl
_FileLineInput1:
   push r25                              ; now read bytes and check End of Line or End of String
   push xh
   push xl
   rcall _FileReadByte                   ; read one byte from file
```

```
   pop xl
   pop xh
   pop r25
   brcs _FileLineInput4                ; error?
   cpi r24, &H0D                       ; CR?
   breq _FileLineInput1                ; ignore &H0D and get next char (should be &H0A)
   cpi r24, &H0A                       ; &H0A is End of Line reached
   breq _FileLineInput2
   tst r25                             ; Just at end of String, do not store any more
   breq _FileLineInput1
   st X+, r24                          ; store byte from file
   dec r25                             ; decr. String-length counter
   rjmp _FileLineInput1                ; next byte
_FileLineInput2:
   clc                                 ; no error sign
   clr r25
_FileLineInput4:
   ldi r24, &H00
   st X, r24                           ; string terminator
 * in r24 , SREG                       ; save C -Flag
   adiw yl, 2                          ; restore Y-pointer from file handle
 * out SREG , r24                      ; restore C -Flag
_FileLineInput5:
   ret
[End]


[_BLoad]
$EXTERNAL _CleanHWStack , _CloseFileHandle , _FileReadByteArray , _GetFreeFileNumber
$EXTERNAL _LoadFileSizeYZ2B , _NormFileName , _OpenFile , _SetDOSError , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; Load a file to SRAM at specified address with length of file
   ; X: Filename
   ; Z: SRAM Address to transfer the file
   ;---------------------------------------------------------------------------
_BLoad:
   push zh                             ; save SRAM Address
   push zl
   rcall _GetFreeFileNumber
   brcs _BLoad3                        ; no free file# ??
   rcall _Normfilename
   brne _BLoad2                        ; wildcard in filename?
 * ldi r25 , cpFileOpenInput
   rcall _OpenFile
   pop xl                              ; SRAM Address for transfer
   pop xh
   brcs _BLoad1
   st -Y, zh                           ; load file handle address to Y0/1
   st -Y, zl
   rcall _LoadFileSizeYZ2B             ; get file size to B
   or r22, r23                         ; is high-word of long zero
   brne _BLoad4                        ; files size >= &H10000
   call _FileReadByteArray0
 * in r24 , SREG
   push r24
   rcall _CloseFileHandle_Y
   adiw yl, 2
   pop r24                             ; adiw yl changes C-Flag
 * out SREG , r24
_BLoad1:
   ret
_BLoad2:
   rcall _SetFileNameWildCardError
_BLoad3:
   rjmp _CleanHWStack2
_BLoad4:
 * ldi r25 , cpFileSizeToGreat         ; set error
   rjmp _SetDOSError_add2toY
[End]
```

```
[_BSave]
$EXTERNAL _CleanHWStack , _CloseFileHandle , _CreateFile , _FileWriteByteArray , _GetFreeFileNumber
$EXTERNAL _NormFileName , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; save a range in SRAM to a file
   ; X: Filename
   ; Z: SRAM Address to transfer
   ; r20, 21 Count of Bytes
   ;---------------------------------------------------------------------------
_BSave:
   push zh                              ; save SRAM Address
   push zl
   push r21                             ; save transfer-length
   push r20
   rcall _GetFreeFileNumber
   brcs _BSave3                         ; no free file# ??
   rcall _Normfilename
   brne _BSave2                         ; wildcard in filename?
 * ldi r25 , cpFileOpenOutput
   rcall _CreateFile_DeleteOld
   pop r20                              ; get length
   pop r21
   pop xl                               ; SRAM Address for transfer
   pop xh
   brcs _BSave1
   st -Y, zh
   st -Y, zl                            ; file handle address
   call _FileWriteByteArray0
 * in r24 , SREG
   push r24
   rcall _CloseFileHandle_Y
   adiw yl, 2
   pop r24                              ; adiw yl changes C-Flag
 * out SREG , r24
_BSave1:
   ret
_BSave2:
   rcall _SetFileNameWildCardError
_BSave3:
   rjmp _CleanHWStack4
[End]


[_FileClose]
$EXTERNAL _ClearDOSError , _CloseFileHandle , _GetFileHandle , _SetErrorFlag
   ;---------------------------------------------------------------------------
   ; Close file, save sector , directory and FAT
   ; Input: File# in r24
   ;---------------------------------------------------------------------------
_FileClose:
   rcall _GetFileHandle
   brcs _FileClose2                     ; file# wrong?
   st -Y, zh                            ; file handle to Y0/1
   st -Y, zl
   rcall _CloseFileHandle_Y             ; do the main part
   brcs _FileClose1                     ; error?
   rjmp _ClearDOSError_Add2toY
_FileClose1:
   rjmp _SetErrorFlag_add2toY
_FileClose2:
   ret
[End]
```

```
[_FileAttr]
$EXTERNAL _ClearDOSError , _GetFileHandle
   ;---------------------------------------------------------------------------
   ; Get the File open mode
   ; Input: r24: file#
   ; output: r24: File open mode
   ;---------------------------------------------------------------------------
_FileAttr:
   rcall _GetFileHandle
   brcs _FileAttr2
 * ldd r24 , Z + co_FileMode             ; load file open mode from file handle area
   rjmp _ClearDOSError
_FileAttr2:
   clr r24                               ; return 0 if error
   ret
[End]


[_FileEOF]
$EXTERNAL _ClearDOSError , _GetFileHandle
   ;---------------------------------------------------------------------------
   ; Get the EOF Status
   ; Input: r24: file#
   ; output: r24: EOF: 0 not EOF; 255 = EOF
   ;---------------------------------------------------------------------------
_FileEOF:
   rcall _GetFileHandle
   brcs _FileEOF2
 * ldd r24 , Z + co_FileBufferStatus
 * andi r24 , dmEOF                      ; mask out EOF -Bit
   breq _FileEOF1                        ; return 0 if not set?
   ser r24                               ; set EOF for return
_FileEOF1:
   rjmp _ClearDOSError
_FileEOF2:
   ser r24                               ; return EOF if error
   ret
[End]


[_FilePutRange]
$EXTERNAL _CheckFileAccess_Mode , _ClearDOSError , _FileWriteByteArray , _GetFileHandle
$EXTERNAL _GotoFilePosition , _SetErrorFlag
   ;---------------------------------------------------------------------------
   ; Put a variable / Byte-array / String to a file
   ; Input: r20/21 Count of bytes (Word-format) to transfer
   ;        r24: file#
   ;        X: Pointer to Variable to transfer
   ;        A (r16-r19) Position (1-based) and T-Flag Set if change to position first
   ;        or T-Flag cleared if transfer from actual position
   ;---------------------------------------------------------------------------
_FilePutRange_8:
   ldi r20, 8
   rjmp _FilePutRange_Bytes              ; for Double
_FilePutRange_4:
   ldi r20, 4                            ; for Long and Single
   rjmp _FilePutRange_Bytes
_FilePutRange_2:
   ldi r20, 2                            ; for Word and Integer
   rjmp _FilePutRange_Bytes
_FilePutRange_1:
   ldi r20, 1
_FilePutRange_Bytes:
   clr r21
_FilePutRange:
   rcall _GetFileHandle
   brcs _FilePutRange3
 * ldi r25 , cFilePut_Mode               ; Mask for File open mode
   rcall _CheckFileAccess_Mode_Z
   brcs _FilePutRange3
   st -Y, zh                             ; filehandle address to Y+0/1
   st -Y, zl
```

```
   brtc _FilePutRange1
   push r21                             ; save length
   push r20
   push xh                              ; save SRAM address
   push xl
   rcall _GotoFilePosition_1Base
   pop xl                               ; get SRAM address
   pop xh
   pop r20                              ; get length
   pop r21
   brcs _FilePutRange2
_FilePutrange1:
   call _FileWriteByteArray0
   brcs _FilePutRange2
   rjmp _ClearDOSError_add2toY
_FilePutRange2:
   rjmp _SetErrorFlag_add2toY
_FilePutRange3:
   ret
[End]


[_FileGetRange]
$EXTERNAL _CheckFileAccess_Mode , _ClearDOSError , _FileReadByteArray , _GetFileHandle
$EXTERNAL _GotoFilePosition , _SetErrorFlag
   ;--------------------------------------------------------------------------------
   ; Get a variable / Byte-array / String from a file
   ; InGet: r20/21 Count of bytes (Word-format) to transfer
   ;        r24: file#
   ;        X: Pointer to Variable to transfer
   ;        A (r16-r19) Position (1-based)  and T-Flag Set if change to position first
   ;        or T-Flag cleared if transfer from actual position
   ;--------------------------------------------------------------------------------
_FileGetRange_8:
   ldi r20, 8                           ; for Double
   rjmp _FileGetRange_Bytes
_FileGetRange_4:
   ldi r20, 4                           ; for Long and Single
   rjmp _FileGetRange_Bytes
_FileGetRange_2:
   ldi r20, 2                           ; for Word and INTEGER
   rjmp _FileGetRange_Bytes
_FileGetRange_1:
   ldi r20, 1
_FileGetRange_Bytes:
   clr r21
_FileGetRange:
   rcall _GetFileHandle
   brcs _FileGetRange3
 * ldi r25 , cFileGet_Mode
   rcall _CheckFileAccess_Mode_Z
   brcs _FileGetRange3
   st -Y, zh                            ; filehandle address to Y+0/1
   st -Y, zl
   brtc _FileGetRange1
   push r21                             ; save length
   push r20
   push xh                              ; save SRAM address
   push xl
   rcall _GotoFilePosition_1Base
   pop xl                               ; get SRAM address
   pop xh
   pop r20                              ; get length
   pop r21
   brcs _FileGetRange2
_FileGetrange1:
   rcall _FileReadByteArray0
   brcs _FileGetRange2
   rjmp _ClearDOSError_add2toY
_FileGetRange2:
   rjmp _SetErrorFlag_add2toY
_FileGetRange3:
   ret
```

```
[End]


[_FileByteInput]
$EXTERNAL _ClearDOSError , _FileReadByte , _GetFileHandle , _SetErrorFlag
   ;---------------------------------------------------------------------------
   ; read one Byte from a file
   ; Input: file# in r24
   ; output: byte in r24
   ;---------------------------------------------------------------------------
_FileByteInput:
   rcall _GetFileHandle
   brcs _FileByteInput3
   st -Y, zh                           ; file handle to Y0/1
   st -Y, zl
_FileByteInput1:
   rcall _FileReadByte
   brcs _FileByteInput2
   rjmp _ClearDOSError_add2toY
_FileByteInput2:
   rcall _SetErrorFlag_add2toY
_FileByteInput3:
   clr r24                             ; return 00 in Byte on Error
   ret
[End]


[_FileByteOutput]
$EXTERNAL _ClearDOSError , _FileWriteByte , _GetFileHandle , _SetErrorFlag
   ;---------------------------------------------------------------------------
   ; write one Byte to the file
   ; Input: file# in r24
   ;        Byte in r25
   ;---------------------------------------------------------------------------
_FileByteOutput:
   push r25
   rcall _GetFileHandle
   pop r24
   brcs _FileByteOutput3
   st -Y, zh                           ; file handle to Y0/1
   st -Y, zl
_FileByteOutput1:
   rcall _FileWriteByte
   brcs _FileByteOutput2
   rjmp _ClearDOSError_add2toY
_FileByteOutput2:
   rjmp _SetErrorFlag_add2toY
_FileByteOutput3:
   ret
[End]


[_FileFlush]
$EXTERNAL _ClearDOSError , _GetFileHandle , _SaveFileSector , _SetBitFileStatusYZ , _SetErrorFlag
   ;---------------------------------------------------------------------------
   ; flush the buffer of a file handle to drive, update Directory and FAT if necessary
   ; Input: file# in r24
   ;---------------------------------------------------------------------------
_FileFlush:
   rcall _GetFileHandle
   brcs _FileFlush2
   st -Y, zh
   st -Y, zl
 * ldi r24 , dmFATDirSaveAnyWay
   rcall _SetBitFileStatusYZ
   rcall _SaveFileSector
   brcs _FileFlush1
   rjmp _ClearDOSError_add2toY
_FileFlush1:
   rjmp _SetErrorFlag_add2toY
```

```
_FileFlush2:
   ret
[End]


[_Dir]
$EXTERNAL _ClearDOSError , _GetFileExtend , _GetFileName , _Mem_Copy , _NormFileName , _SearchDirEntry
$EXTERNAL _SearchNextDirEntry
$EXTERNAL _RTRIM
   ;--------------------------------------------------------------------------
   ; search a file on the drive (_DIR) or search next matching file (_DIR0) to
   ; a previous _DIR command
   ; input: FileName in X  (_Dir) or none with _Dir0
   ; return file name string: Address in Z
   ;--------------------------------------------------------------------------
_Dir:
   push zh
   push zl
   rcall _NormFileName                  ; brings Filename to Z
   rcall _SearchDirEntry
   brcs _Dir1
   push zh
   push zl
   loadadr gsTempFileName , Z           ; copy Dir info of last found file
   loadadr gsDir0TempFileName , X       ; for next DIR()
   ldi r24, 12                          ; copy TempFileName
   rcall _Mem_Copy
   pop zl
   pop zh
   clc
_Dir1:
   pop xl
   pop xh
   brcc _Dir2
   clr r24
   st X , r24                           ; empty String
 * cpi r25 , cpFileNotFound
   brne _Dir3
   rjmp _ClearDOSError
_Dir3:
   ret
_Dir2:
   rcall _GetFilename
  call _RTrim0
   sbiw xl, 1
   ldi r24, asc(".")
   st X+, r24
   rcall _GetFileExtend
   call _RTrim0
   loadadr gwDirEntry , Z
   loadadr gwDir0Entry , X
   ldi r24, 6                           ; copy gwDirEntry, glDirSectorNumber
   rjmp _Mem_Copy

   ; entry without filename
_Dir0:
   push zh
   push zl
   rcall _SearchNextDirEntry
   rjmp _Dir1
[end]
```

```
[_FileLen]
$EXTERNAL _A2LongX , _ClearA , _GetDirEntry , _GetFileSize , _SearchDirEntry
   ;---------------------------------------------------------------------------
   ; get the file-size of a file on the drive (_FileLen) searched by a file name
   ; without wildcard or of a previous searched file with _Dir/Dir0
   ; input: FileName in X  (_FileLen) or none with _FileLen0
   ;         Z-Pointer set at Long-Variable for result
   ; return File-Len written to Z (Long)
   ;---------------------------------------------------------------------------
_FileLen:
   push zh
   push zl
   rcall _SearchDirEntry_CheckFileName    ; with norming file name and checking wildcard
   rjmp _FileLen1
_FileLen0:
   push zh
   push zl
   rcall _GetDirEntry_0
_FileLen1:
   pop xl
   pop xh
   brcc _FileLen2
   rcall _ClearA
   rcall _A2LongX
   ret
_FileLen2:
   rjmp _GetFileSize
[end]


[_FileDateTimeB]
$EXTERNAL _GetDirEntry , _GetFileDate , _GetFileTime , _SearchDirEntry
   ;---------------------------------------------------------------------------
   ; get the file Date-time in intern Clock-format of a file on the drive (_FileDateTimeB)
   ; searched by a file name without wildcard or of a previous searched file with _Dir/Dir0
   ; input: FileName in X  (_FilDateTimeB) or none with _FileDateTimeB0
   ;         Z-Pointer set to 6-byte array with sec/min/hour/day/month/Year
   ; return Date/Time written to Z (6 Bytes)
   ;---------------------------------------------------------------------------
_FileDateTimeB:
   push zh
   push zl
   rcall _SearchDirEntry_CheckFileName    ; with norming file name and checking wildcard
   rjmp _FileDateTimeB1
_FileDateTimeB0:
   push zh
   push zl
   rcall _GetDirEntry_0
_FileDateTimeB1:
   pop xl
   pop xh
   brcc _FileDateTimeB2
   ret
_FileDateTimeB2:
   rcall _GetFileTime
   rjmp _GetFileDate
[end]


[_FileDateTimeS]
$EXTERNAL _GetDirEntry , _GetFileDate , _GetFileTime , _SearchDirEntry
$EXTERNAL _Clock2Date , _Clock2Time
   ;---------------------------------------------------------------------------
   ; get the file Date-time of a file as String on the drive (_FileDateTimeS)
   ; searched by a file name without wildcard or of a previous searched file with
   ; _Dir/Dir0 (_FileDateTimeS0)
   ; input: FileName in X  (_FileDateTimeS) or none with _FileDateTimeS0
   ;         Z-Pointer set at String-Variable for result
   ; return File-Date-Time written to Z (String with 17 Bytes)
   ;---------------------------------------------------------------------------
_FileDateTimeS:
   push zh
```

```
   push zl
   rcall _SearchDirEntry_CheckFileName    ; with norming file name and checking wildcard
   rjmp _FileDateTimeS1
_FileDateTimeS0:
   push zh
   push zl
   rcall _GetDirEntry_0
_FileDateTimeS1:
   brcc _FileDateTimeS2
   pop xl
   pop xh
   clr r24
   st X, r24                              ; String terminator to return value
   ret
_FileDateTimeS2:
   ldi xl, 10                             ; store time and date (Byte-Format) to r10-r15
   clr xh
   rcall _GetFileTime
   rcall _GetFileDate
   pop xl
   pop xh
  call _ClockReg2Date
   ldi r24, 32                            ; seperate with blank
   sbiw xl, 1                             ; go back to string terminator
   st X+, r24                             ; overwrite string terminator with blank
   ldi zl, 10                             ; switch to time r10-r12
   jmp _Clock2Time
[End]


[_FileDateS]
$EXTERNAL _GetDirEntry , _GetFileDate , _SearchDirEntry
$EXTERNAL _Clock2Date
   ;--------------------------------------------------------------------------------
   ; get the file Date of a file as String on the drive (_FileDateS)
   ; searched by a file name without wildcard or of a previous searched file with
   ; _Dir/Dir0 (_FileDateS0)
   ; input: FileName in X  (_FileDateS) or none with _FileDateS0
   ;         Z-Pointer set at String-Variable for result
   ; return File-Date written to Z (String with 8 Bytes)
   ;--------------------------------------------------------------------------------
_FileDateS:
   push zh
   push zl
   rcall _SearchDirEntry_CheckFileName    ; with norming file name and checking wildcard
   rjmp _FileDateS1
_FileDateS0:
   push zh
   push zl
   rcall _GetDirEntry_0
_FileDateS1:
   brcc _FileDateS2
   pop xl
   pop xh
   clr r24
   st X, r24                              ; String terminator to return value
   ret
_FileDateS2:
   ldi xl, 13                             ; store time and date (Byte-Format) to r10-r15
   clr xh
   rcall _GetFileDate
   pop xl
   pop xh
   jmp _ClockReg2Date
[End]
```

```
[_FileTimeS]
$EXTERNAL _GetDirEntry , _GetFileTime , _SearchDirEntry
$EXTERNAL _Clock2Time
   ;--------------------------------------------------------------------------------
   ; get the file time of a file as String on the drive (_FileTimeS)
   ; searched by a file name without wildcard or of a previous searched file with
   ; _Dir/Dir0 (_FileTimeS0)
   ; input: FileName in X  (_FileTimeS) or none with _FileTimeS0
   ;         Z-Pointer set at String-Variable for result
   ; return File-Time written to Z (String with 8 Bytes)
   ;--------------------------------------------------------------------------------
_FileTimeS:
   push zh
   push zl
   rcall _SearchDirEntry_CheckFileName    ; with norming file name and checking wildcard
   rjmp _FileTimeS1
_FileTimeS0:
   push zh
   push zl
   rcall _GetDirEntry_0
_FileTimeS1:
   brcc _FileTimeS2
   pop xl
   pop xh
   clr r24
   st X, r24                              ; String terminator to return value
   ret
_FileTimeS2:
   ldi xl, 13                             ; store time and date (Byte-Format) to r10-r15
   clr xh
   rcall _GetFileTime
   pop xl
   pop xh
   jmp _ClockReg2Time
[End]


[_GetAttr]
$EXTERNAL _GetDirEntry , _GetFileAttribute , _SearchDirEntry
   ;--------------------------------------------------------------------------------
   ; get the file attribut of a file as Byte on the drive (_GetAttr)
   ; searched by a file name without wildcard or of a previous searched file with
   ; _Dir/Dir0 (_GetAttr0)
   ; input: FileName in X  (_GetAttr) or none with _GetAttr0
   ;         Z-Pointer set at Byte-Variable for result
   ; return File-attribut written to Z (Byte)
   ;--------------------------------------------------------------------------------
_GetAttr:
   push zh
   push zl
   rcall _SearchDirEntry_CheckFileName    ; with norming file name and checking wildcard
   rjmp _GetAttr1
_GetAttr0:
   push zh
   push zl
   rcall _GetDirEntry_0
_GetAttr1:
   pop xl
   pop xh
   brcc _GetAttr2
   clr r24
   st X, r24
   ret
_GetAttr2:
   rjmp _GetFileAttribute
[end]
```

```
[_GetNextSectorNumber]
$EXTERNAL _Add32_Word , _Cluster2Sector , _GetNextCluster , _Sector2Cluster , _Sub32_1
   ;---------------------------------------------------------------------------
   ; checks whether next Sector to sector# in A will overflow current cluster
   ; Input A: current sector#
   ; Output A´: next sector#
   ; Zero flag set if no overflow
   ;---------------------------------------------------------------------------
_GetNextSectorNumber:
   mov r24, r16
   rcall _add32_1                       ; Sector# + 1
   lds r25, {glDataFirstSector}         ; get LSB of first Data Sector#
   sub r24 , r25                        ; build offset
   mov r25, r24
   inc r24                              ; +1 = simulate next sector#
   eor r24, r25                         ; set changed bits to 1
   lds r25, {gbSectorsPerCluster}       ; f.E. 00000100 for 4 Sectors per Cluster
   and r24, r25                         ; check overflow to significant bit(bit#=2 in examples)
   brne _GetNextSectorNumber2
_GetNextSectorNumber1:
   clc                                  ; no cluster overflow - clear Error Flag
   ret
_GetNextSectorNumber2:
   rcall _Sub32_1                       ; cluster overflow, go back to previous sector#
   rcall _Sector2Cluster                ; get old cluster#
   rcall _GetNextCluster
   brcc _GetNextSectorNumber3
   ret                                  ; give error back, if no next cluster found
_GetNextSectorNumber3:
   rcall _Cluster2Sector                ; conver new cluster# to first sector# in cluster
   rjmp _GetNextSectorNumber1
[end]


[_Sector2Cluster]
$EXTERNAL _LongZ2B
$EXTERNAL _SUB32
   ;---------------------------------------------------------------------------
   ; Convert sector# in A to Cluster# in A
   ;---------------------------------------------------------------------------
_Sector2Cluster:
   loadadr glDataFirstSector , Z
   rcall _LongZ2B
   call _Sub32                          ; sector offset between Current Sector# and Data first Sector
   brmi _Sector2Cluster3                ; Sector# is lower than first data sector: error
   lds r24, {gbSectorsPerCluster}
_Sector2Cluster1:
   sbrc r24, 0
   rjmp _Sector2Cluster2
   lsr r19
   ror r18
   ror r17
   ror r16
   lsr r24
   rjmp _Sector2Cluster1
_Sector2Cluster2:
   subi r16, -2                         ; + 2 because first cluster has number 2
   sbci r17, -1
   sbci r18, -1
   sbci r19, -1                         ; must always result in Plus condition
_Sector2Cluster3:
   ret
[end]
```

```
[_Cluster2Sector]
$EXTERNAL _LongZ2B
$EXTERNAL _ADD32
   ;--------------------------------------------------------------------------------
   ; calculate from cluster# the first sector# of this cluster
   ; Input:    A (r16-19): cluster#
   ; Output:   A (r16-19): sector#
   ;--------------------------------------------------------------------------------
_Cluster2Sector:
   lds r24, {gbSectorsPerCluster}
   subi r16, 2                          ; - 2 because first cluster is 2
   sbci r17, 0
   sbci r18, 0
   sbci r19, 0
_Cluster2Sector1:
   sbrc r24, 0                          ; multiply cluster# with sectors per cluster
   rjmp _Cluster2Sector2                ; with shifting left
   lsl r16
   rol r17
   rol r18
   rol r19
   lsr r24
   rjmp _Cluster2Sector1
_Cluster2Sector2:
   loadadr glDataFirstSector , Z
   rcall _LongZ2B                       ; add sector# of first data sector to sector offset
   jmp _Add32                           ; from cluster# * sectors per cluster
[end]


[_GetNextCluster]
$EXTERNAL _ClearDOSError , _GetSector , _LoadFileSectorNumberYZ2A , _LongX2B , _Sector2Cluster
$EXTERNAL _SetDOSError
$EXTERNAL _ADD32
   ;--------------------------------------------------------------------------------
   ; Get next Cluster to a given cluster from the FAT
   ; Input: Cluster# in r16-r19 (A)
   ; Output: Cluster# in r16-r19 (A), T-Flag set according to FAT16/32
   ;--------------------------------------------------------------------------------
_GetNextCluster_Y:
   rcall _LoadFileSectorNumberYZ2A
   rcall _Sector2Cluster
_GetNextCluster:
   loadadr glFatFirstSector , X         ; FAT -Begin
   rcall _LongX2B
   ; High Byte is Sector-Offset in FAT-Area, Low Byte * 2 is Offset in Sector
   lds r0, {gbFileSystemStatus}         ; Info about FAT16/32
 * sbrs r0 , dFileSystemStatusFAT
   rjmp _GetNextCluster1
   lsl r16
   rol r17
   rol r18
   rol r19
_GetNextCluster1:
   push r16                             ; LSB is offset in FAT Sector
   mov r16, r17                         ; MSB is offset to FATFirstSector
   mov r17, r18
   mov r18, r19
   clr r19
   call _add32
   rcall _GetSector_FAT
   ; now Z points to buffer
   pop r24
   brcs _GetNextCluster5                ; set DOS-Error once more, but saves Code
   lds r0, {gbFileSystemStatus}         ; Info about FAT16/32
 * bst r0 , dFileSystemStatusFAT
   clr r25
   lsl r24                              ; Low-Byte * 2 = Buffer-Offset
   rol r25
   add zl, r24                          ; add to bufferstart
   adc zh, r25
   clr r18                              ; clear for FAT16
   clr r19
```

```
   ld r16, Z+                          ; Load new Cluster address
   ld r17, Z+
   brtc _GetNextCluster2               ; FAT16/32
   ld r18, Z+
   ld r19, Z+
   andi r19, &H0F                      ; mask bit 28-31
   sbiw zl, 2                          ; Set Z back to Cluster Entry Part1 of FAT32
   ldi r22, &HFF                       ; FAT32 High Word
   ldi r23, &H0F
_GetNextCluster2:
   sbiw zl, 2                          ; set Z back to Cluster Entry for FAT16 and FAT32 Part2
   ldi r20, &HF8
   ldi r21, &HFF
   cp r16, r20                         ; check for last cluster of File
   cpc r17, r21
   brtc _GetNextCluster4
   cpc r18, r22                        ; high Word for FAT32
   cpc r19, r23
_GetNextCluster4:
   brcs _GetNextCluster6
 * ldi r25 , cpNoNextCluster
_GetNextCluster5:
   rjmp _SetDosError
_GetNextCluster6:
   rjmp _ClearDOSError
[end]


[_GetFreeCluster]
$EXTERNAL _A2LongX , _Add32_Word , _ClearA , _ClearDOSError , _GetNextCluster , _LongX2A
$EXTERNAL _SetDOSError
   ;--------------------------------------------------------------------------------
   ; Search Free Cluster in FAT (00 00)
   ; Count Free Clusters at GetFreeCluster0
   ; OUTPUT: cluster# in A (r16, r17, r18, r19)
   ; Output: Count of free Clusters in B
   ;--------------------------------------------------------------------------------
   ; entry for counting Free Clusters
_GetFreeCluster0:
   rcall _ClearA
   ldi r16, 2
   mov r10, r16                        ; sign for Counting free cluster (bit1 = 1)
   clr r20                             ; count of free cluster in B
   clr r21
   clr r22
   clr r23
   rjmp _GetFreeCluster1
_GetFreeCluster:
   Loadadr glLastSearchedCluster , X   ; start at last found free cluster entry
   rcall _LongX2A
   clr r10
_GetFreeCluster1:
   push r19
   push r18
   push r17
   push r16
   push r23
   push r22
   push r21
   push r20
   push r10
   rcall _GetNextCluster
   pop r10
   pop r20
   pop r21
   pop r22
   pop r23
   pop r16
   pop r17
   pop r18
   pop r19
   loadadr glMaxClusterNumber , X      ; load highest possible cluster#
   ld r12, X+
```

```
   ld r13, X+
   ld r14, X+
   ld r15, X+
   brcc _GetFreeCluster2                ; Cluster# found or free
 * cpi r25 , cpNoNextCluster            ; check for error , cpNoNextCluster is here OK
   brne _GetFreeCluster10               ; other error in fetching Sector go exit
_GetFreeCluster2:
   cp r12, r16
   cpc r13, r17
   cpc r14, r18
   cpc r15, r19
   brcs _GetFreeCluster9                ; cluster# overflow
   clr r25
   ld r24, Z+                           ; Load new Cluster address
   or r25, r24
   ld r24, Z+
   or r25, r24
   brtc _GetFreeCluster3
   ld r24, Z+
   or r25, r24
   ld r24, Z+
   andi r24, &H0F
   or r25, r24
_GetFreeCluster3:
   brne _GetFreeCluster4
   sbrs r10, 1                          ; counting free cluster?
   rjmp _GetFreeCluster6
   subi r20, -1                         ; +1, increase count of free clusters
   sbci r21, -1
   sbci r22, -1
   sbci r23, -1
_GetFreeCluster4:
   mov r24, r16                         ; save info for cluster# overflow i sector
   mov r25, r17
   rcall _Add32_1                       ; Incr Cluster #
   eor r25, r17                         ; FAT16 FAT-Sector Overflow if Bit0 = 1, bit8 toggled
   lsr r25                              ; toggled bit to carry
   brtc _GetFreeCluster5                ; branch if FAT16
   eor r24, r16                         ; FAT32 Fat-Sector overflow if bit7 = 1, bit7 toggled
   lsl r24                              ; to Carry-bit
_GetFreeCluster5:
   brcc _GetFreeCluster2                ; no overflow, check next cluster# in sector
   rjmp _GetFreeCluster1                ; load next Sector to cluster# in A
_GetFreeCluster6:
   sbiw zl, 2                           ; Z-> Cluster-Entry
   brtc _GetFreeCluster7                ; FAT32?
   sbiw zl, 2
_GetFreeCluster7:
   loadadr glLastSearchedCluster , X
   rcall _A2LongX                       ; save last searched Cluster#
   rjmp _ClearDOSError
_GetFreeCluster9:
 * ldi r25 , cpNoFreeCluster
_GetFreeCluster10:
   rjmp _SetDOSError
[end]
```

```
   ;*** H e l p e r   R o u t i n e s ****************************************

[_ClearA]
   ;--------------------------------------------------------------------------------
   ; Clear Accu A (r16-19)
   ;--------------------------------------------------------------------------------
_ClearA:
   clr r16
   clr r17
   clr r18
   clr r19
   ret
[end]


[_Add32_Word]
$EXTERNAL _ADD32
   ;--------------------------------------------------------------------------------
   ; add to a long in A (r16-r19) numeric 1, a byte (r20) or a word (r20/r21)
   ; Input: Long in A
   ;         Byte in r20 _Add32_Byte
   ;         Word in r20/r21 _Add32_Word
   ; Result in A (r16-r19)
   ;--------------------------------------------------------------------------------
_Add32_1:
   subi r16, -1                         ; +1
   sbci r17, -1
   sbci r18, -1
   sbci r19, -1
   ret
_Add32_Byte:
   clr r21
_Add32_Word:
   clr r22
   clr r23
_Add32_r:
   jmp _Add32
[end]


[_Sub32_1]
   ;--------------------------------------------------------------------------------
   ; subtract 1 from Long in A (r16-r19)
   ; Input: Long in A (r16-r19)
   ; result Long in A
   ;--------------------------------------------------------------------------------
_Sub32_1:
   subi r16, 1
_Sub32_Byte:
   sbci r17, 0
_Sub32_Word:
   sbci r18, 0
   sbci r19, 0
   ret
[end]


[_LongX2A]
   ;--------------------------------------------------------------------------------
   ; Load Long from SRAM to A
   ; Pointer to Long in X
   ;--------------------------------------------------------------------------------
_LongX2A:
   ld r16, X+
   ld r17, X+
   ld r18, X+
   ld r19, X+
   ret
[end]
```

```
[_LongZ2A]
    ;-----------------------------------------------------------------------------
    ; Load Long from SRAM to A
    ; Pointer to Long in Z
    ;-----------------------------------------------------------------------------
_LongZ2A:
    ld r16, Z+
    ld r17, Z+
    ld r18, Z+
    ld r19, Z+
    ret
[end]


[_LongX2B]
    ;-----------------------------------------------------------------------------
    ; Load Long from SRAM to B
    ; Pointer to Long in X
    ;-----------------------------------------------------------------------------
_LongX2B:
    ld r20, X+
    ld r21, X+
    ld r22, X+
    ld r23, X+
    ret
[end]


[_LongZ2B]
    ;-----------------------------------------------------------------------------
    ; Load Long from SRAM to B
    ; Pointer to Long in Z
    ;-----------------------------------------------------------------------------
_LongZ2B:
    ld r20, Z+
    ld r21, Z+
    ld r22, Z+
    ld r23, Z+
    ret
[end]


[_A2LongX]
    ;-----------------------------------------------------------------------------
    ; Load Long from A to SRAM
    ; Pointer to Long in X
    ;-----------------------------------------------------------------------------
_A2LongX:
    st x+, r16
    st x+, r17
    st x+, r18
    st x+, r19
    ret
[end]


[_A2LongZ]
    ;-----------------------------------------------------------------------------
    ; Load Long from A to SRAM
    ; Pointer to Long in Z
    ;-----------------------------------------------------------------------------
_A2LongZ:
    st Z+, r16
    st Z+, r17
    st Z+, r18
    st Z+, r19
    ret
[end]
```

```
[_B2LongZ]
   ;---------------------------------------------------------------------------
   ; Load Long from B to SRAM
   ; Pointer to Long in Z
   ;---------------------------------------------------------------------------
_B2LongZ:
   st Z+, r20
   st Z+, r21
   st Z+, r22
   st Z+, r23
   ret
[end]


[_WordZ2A]
   ;---------------------------------------------------------------------------
   ; Load Word from SRAM to A (r16,r17)
   ; Pointer to Word in Z
   ;---------------------------------------------------------------------------
_WordZ2A_L:
   clr r18
   clr r19
_WordZ2A:
   ld r16, Z+
   ld r17, Z+
   ret
[end]


[_WordX2B]
   ;---------------------------------------------------------------------------
   ; Load Word from SRAM to A (r16,r17)
   ; Pointer to Word in X
   ;---------------------------------------------------------------------------
_WordX2B_L:
   clr r22
   clr r23
_WordX2B:
   ld r20, X+
   ld r21, X+
   ret
[end]


[_WordZ2B]
   ;---------------------------------------------------------------------------
   ; Load Word from SRAM to B (r16,r17)
   ; Pointer to Word in Z
   ;---------------------------------------------------------------------------
_WordZ2B:
   ld r20, Z+
   ld r21, Z+
   ret
[end]


[_B2WordX]
   ;---------------------------------------------------------------------------
   ; Load Word from B to SRAM
   ; Pointer to Word in X
   ;---------------------------------------------------------------------------
_B2WordX:
   st X+, r20
   st X+, r21
   ret
[end]
```

```
[_A2WordX]
   ;---------------------------------------------------------------------------
   ; Load Word from A to SRAM
   ; Pointer to Word in X
   ;---------------------------------------------------------------------------
_A2WordX:
   st X+, r16
   st X+, r17
   ret
[end]


[_B2WordZ]
   ;---------------------------------------------------------------------------
   ; Load Word from B to SRAM
   ; Pointer to Word in Z
   ;---------------------------------------------------------------------------
_B2WordZ:
   st Z+, r20
   st Z+, r21
   ret
[end]


[_Mov4A2B]
   ;---------------------------------------------------------------------------
   ; Move Long from A to B
   ;---------------------------------------------------------------------------
_Mov4A2B:
   mov r20, r16
   mov r21, r17
   mov r22, r18
   mov r23, r19
   ret
[end]


[_CheckA_Zero]
   ;---------------------------------------------------------------------------
   ; Checks whether A (r16-19) contains 0
   ; Output: Z-Falg set if true
   ;---------------------------------------------------------------------------
_CheckA_Zero:
   subi r16, 0
   sbci r17, 0
   sbci r18, 0
   sbci r19, 0
   ret
[end]


[_Comp16_r20_r22]
   ;---------------------------------------------------------------------------
   ; Compare Word in r20,r21 with r22,r23
   ; Output: Flags set according cp (r20-21), (r22-23)
   ;---------------------------------------------------------------------------
_Comp16_r20_r22:
   cp r20, r22
   cpc r21, r23
   ret
[end]


[_ShiftR_WordB]
   ;---------------------------------------------------------------------------
   ; Shifts Word in B number of bits to right
   ; Value in r20, r21
   ; number of shifts in r24
```

```
   ;---------------------------------------------------------------------------
_ShiftR_WordB:
   lsr r21
   ror r20
   dec r24
   brne  _SHIFTR_WORDB
   ret
[end]


[_ShiftL_WordB]
   ;---------------------------------------------------------------------------
   ; Shifts Word in B number of bits to left
   ; Value in r20, r21
   ; number of shifts in r24
   ;---------------------------------------------------------------------------
_ShiftL_WordB:
   lsl r20
   rol r21
   dec r24
   brne  _SHIFTL_WORDB
   ret
[end]



[_Mem_Copy]
   ;---------------------------------------------------------------------------
   ; copy bytes from (Z) to (X)
   ; count of bytes in r24
   ;---------------------------------------------------------------------------
_Mem_Copy_4:
   ldi r24, 4                          ; copy 4 Byte at _mem_Copy_4
_Mem_Copy:
   push r25
_Mem_Copy1:
   ld r25, z+
   st x+, r25
   dec r24
   brne  _Mem_Copy1
   pop r25
   ret
[end]



[_AddrFileHandleBase2Z]
   ;---------------------------------------------------------------------------
   ; load Z with file handle base address from Y+0/1
   ;---------------------------------------------------------------------------
_AddrFileHandleBase2Z:
   ld zl, Y+0
   ld zh, Y+1
   ret
[end]


[_AddrFileFirstCluster2X]
   ;---------------------------------------------------------------------------
   ; Load X with address of FileFirstCluster of current file-handle
   ;---------------------------------------------------------------------------
_AddrFileFirstCluster2X:
   ld xl, Y+0
   ld xh, Y+1
 * adiw xl , co_FileFirstCluster
   ret
[end]


[_AddrFileSize2Z]
   ;---------------------------------------------------------------------------
```

```
   ; Load Z with Address of FileSize of current file-handle
   ;-----------------------------------------------------------------------------
_AddrFileSize2Z:
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileSize
   ret
[end]


[_SaveFileDirSectorNumberA2YZ]
$EXTERNAL _A2LongZ
   ;-----------------------------------------------------------------------------
   ; Save FileDirSectorNumber to current file-handle from A (using Z-Pointer)
   ;-----------------------------------------------------------------------------
_SaveFileDirSectorNumberA2YZ:
   ldd zl, Y+0
   ldd zh, Y+1
 * adiw zl , co_FileDirSectorNumber
   rjmp _A2LongZ
[end]


[_SaveFileFirstClusterA2YZ]
$EXTERNAL _A2LongZ
   ;-----------------------------------------------------------------------------
   ; Save FileFirstCluster to current file-handle from A (using Z-Pointer)
   ;-----------------------------------------------------------------------------
_SaveFileFirstClusterA2YZ:
   ldd zl, Y+0
   ldd zh, Y+1
 * adiw zl , co_FileFirstCluster
   rjmp _A2LongZ
[end]


[_LoadFileFirstClusterYZ2A]
   ;-----------------------------------------------------------------------------
   ; Load A with FileFirstCluster from current file-handle (using Z-Pointer)
   ;-----------------------------------------------------------------------------
$EXTERNAL _LongZ2A
_LoadFileFirstClusterYZ2A:
   ldd zl, Y+0
   ldd zh, Y+1
 * adiw zl , co_FileFirstCluster
   rjmp _LongZ2A
[end]


[_SaveFileSizeA2YZ]
$EXTERNAL _A2LongZ , _AddrFileSize2Z
   ;-----------------------------------------------------------------------------
   ; Save FileSize to current file-handle from A (using Z-Pointer)
   ;-----------------------------------------------------------------------------
_SaveFileSizeA2YZ:
   rcall _AddrFileSize2Z
   rjmp _A2LongZ
[end]


[_LoadFileSizeYZ2A]
$EXTERNAL _AddrFileSize2Z , _LongZ2A
   ;-----------------------------------------------------------------------------
   ; Load A with FileSize from current file-handle (using Z-Pointer)
   ;-----------------------------------------------------------------------------
_LoadFileSizeYZ2A:
   rcall _AddrFileSize2Z
   rjmp _LongZ2A
[end]
```

```
[_LoadFileSizeYZ2B]
$EXTERNAL _AddrFileSize2Z , _LongZ2B
   ;---------------------------------------------------------------------------
   ; Load B with FileSize from current file-handle (using Z-Pointer)
   ;---------------------------------------------------------------------------
_LoadFileSizeYZ2B:
   rcall _AddrFileSize2Z
   rjmp _LongZ2B
[end]


[_AddrFileSize2X]
   ;---------------------------------------------------------------------------
   ; Load X with address of FileSize of current file-handle
   ;---------------------------------------------------------------------------
_AddrFileSize2X:
   ld xl, Y+0
   ld xh, Y+1
 * adiw xl , co_FileSize
   ret
[end]


[_AddrFileSectorNumber2Z]
   ;---------------------------------------------------------------------------
   ; Load Z with address of FileSectorNumber of current file-handle
   ;---------------------------------------------------------------------------
_AddrFileSectorNumber2Z:
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileSectorNumber
   ret
[end]


[_LoadFileSectorNumberYZ2A]
$EXTERNAL _AddrFileSectorNumber2Z , _LongZ2A
   ;---------------------------------------------------------------------------
   ; Load A with FileSectorNumber from current file-handle (using Z-Pointer)
   ;---------------------------------------------------------------------------
_LoadFileSectorNumberYZ2A:
   rcall _AddrFileSectorNumber2Z
   rjmp _LongZ2A
[end]


[_AddrFileBufferStatus2Z]
   ;---------------------------------------------------------------------------
   ; Load Z with address of FileBufferStatus of current file-handle
   ;---------------------------------------------------------------------------
_AddrFileBufferStatus2Z:
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileBufferStatus
   ret
[end]



[_LoadFileBufferStatusYZ]
   ;---------------------------------------------------------------------------
   ; Load Z with address of FileBufferStatus of current file-handle
   ;---------------------------------------------------------------------------
_LoadFileBufferStatusYZ:
   ld zl, Y+0
   ld zh, Y+1
 * ld _temp1 , Z + co_FileBufferStatus
```

```
   ret
[end]


[_SaveFileBufferStatusYZ]
   ;---------------------------------------------------------------------------
   ; Save _temp1 (r24) to FileBufferStatus of current file-handle (using Z-Pointer)
   ;---------------------------------------------------------------------------
_SaveFileBufferStatusYZ:
   ld zl, Y+0
   ld zh, Y+1
 * std Z + co_FileBufferStatus , _temp1
   ret
[end]


[_AddrFilePosition2Z]
   ;---------------------------------------------------------------------------
   ; Load Z with Address of FilePosition of current file-handle
   ;---------------------------------------------------------------------------
_AddrFilePosition2Z:
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FilePosition
   ret
[end]


[_LoadFilePositionYZ2A]
$EXTERNAL _AddrFilePosition2Z , _LongZ2A
   ;---------------------------------------------------------------------------
   ; Load A with FilePosition from current file-handle (using Z-Pointer)
   ;---------------------------------------------------------------------------
_LoadFilePositionYZ2A:
   rcall _AddrFilePosition2Z
   rjmp _LongZ2A
[end]


[_LoadFilePositionYZ2B]
$EXTERNAL _AddrFilePosition2Z , _LongZ2B
   ;---------------------------------------------------------------------------
   ; Load B with FilePosition from current file-handle (Using Z-Pointer)
   ;---------------------------------------------------------------------------
_LoadFilePositionYZ2B:
   rcall _AddrFilePosition2Z
   rjmp _LongZ2B
[end]


[_SaveFilePositionA2YZ]
$EXTERNAL _A2LongZ , _AddrFilePosition2Z
   ;---------------------------------------------------------------------------
   ; Save A to FilePosition of current file-handle (using Z-Pointer)
   ;---------------------------------------------------------------------------
_SaveFilePositionA2YZ:
   rcall _AddrFilePosition2Z
   rjmp _A2LongZ
[end]


[_SaveFilePositionB2YZ]
$EXTERNAL _AddrFilePosition2Z , _B2LongZ
   ;---------------------------------------------------------------------------
   ; Save B to FilePosition of current file-handle (Using Z-Pointer)
   ;---------------------------------------------------------------------------
_SaveFilePositionB2YZ:
```

```
   rcall _AddrFilePosition2Z
   rjmp _B2LongZ
[end]


[_AddrFilePosition2X]
   ;---------------------------------------------------------------------------
   ; Load X with address of FilePosition of current file-handle
   ;---------------------------------------------------------------------------
_AddrFilePosition2X:
   ld xl, Y+0
   ld xh, Y+1
 * adiw xl , co_FilePosition
   ret
[end]


[_LastCluster2FAT]
$EXTERNAL _SetDirFATWritePending
   ;---------------------------------------------------------------------------
   ; Write last cluster# ( 0F FF / FF FF) to FAT-Buffer
   ; Input: Z points to current Cluster-entry in FAT-buffer
   ; FAT16 or 32 is considered
   ;---------------------------------------------------------------------------
_LastCluster2FAT:
   lds r0, {gbFileSystemStatus}
 * bst r0 , dFileSystemStatusFAT
_LastCluster2FATT:
   ldi r24, &HFF
   st Z+, r24                          ; set to last Cluster#
   st Z+, r24
   brtc _LastCluster2FAT2
   st Z+, r24
   ldi r24, &H0F
   st Z+, r24
_LastCluster2FAT2:
   rjmp _SetFATWritePending
[end]


[_ClusterA2FAT]
$EXTERNAL _SetDirFATWritePending
   ;---------------------------------------------------------------------------
   ; Write cluster# from A (r16-17 for FAT16; r16-19 for FAT32) to FAT-Buffer
   ; Input: Z points to current Cluster-entry in FAT-buffer
   ; FAT16 or 32 is considered
   ;---------------------------------------------------------------------------
_ClusterA2FAT:
   lds r0, {gbFileSystemStatus}
 * bst r0, dFileSystemStatusFAT
   st Z+, r16
   st Z+, r17
   brtc _ClusterA2FAT1
   st Z+, r18
   st Z+, r19
_ClusterA2FAT1:
   rjmp _SetFATWritePending
[end]




[_AddrFileBuffer2X]
   ;---------------------------------------------------------------------------
   ; load pointer to sector-buffer of a filehandle to X
   ; Input: Y+0/1: File handle
   ;---------------------------------------------------------------------------
_AddrFileBuffer2X:
   ldd xl, Y+0
   ldd xh, Y+1
```

```
 * adiw xl , co_FileBuffer
   ret
[end]


[_AddrFileBuffer2Z]
   ;---------------------------------------------------------------------------
   ; load pointer to sector-buffer of a filehandle to Z
   ; Input: Y+0/1: File handle
   ;---------------------------------------------------------------------------
_AddrFileBuffer2Z:
   ldd zl, Y+0
   ldd zh, Y+1
 * adiw Zl , co_FileBuffer
   ret
[end]


[_SaveFileSectorNumberA2YZ]
$EXTERNAL _A2LongZ , _AddrFileSectorNumber2Z
   ;---------------------------------------------------------------------------
   ; Save A to FileSectorNumber of current file-handle (using Z-Pointer)
   ;---------------------------------------------------------------------------
_SaveFileSectorNumberA2YZ:
   rcall _AddrFileSectorNumber2Z
   rjmp _A2LongZ
[end]


[_AddrFileSectorNumber2X]
   ;---------------------------------------------------------------------------
   ; Load X with address of FileSectorNumber of current file-handle
   ;---------------------------------------------------------------------------
_AddrFileSectorNumber2X:
   ld xl, Y+0
   ld xh, Y+1
 * adiw xl , co_FileSectorNumber
   ret
[end]


[_CleanHWStack]
   ;---------------------------------------------------------------------------
   ; clean HW,stack if push values are not needed any more, in case of error
   ; gets 1 to 6 bytes from stack
   ; Stack behind cleaning range must contain a valid RETURN address
   ;---------------------------------------------------------------------------
_CleanHWStack6:
   pop r0
_CleanHWStack5:
   pop r0
_CleanHWStack4:
   pop r0
_CleanHWStack3:
   pop r0
_CleanHWStack2:
   pop r0
_CleanHWStack1:
   pop r0
   ret
[end]
```

```
   ;*** E R R O R   H a n d l i n g ********************************************

[_SetDOSError]
   ;-------------------------------------------------------------------------------
   ; write Error code to gbDOSError and set Carry-Flag
   ; Input: Error-code in r25
   ;-------------------------------------------------------------------------------
_SetDOSError_add2toY:
   adiw yl, 2                           ; adjust Y-Pointer
_SetDOSError:
   sec                                  ; C = Error-Flag
_SetDOSError1:
   sts {gbDOSError}, r25                ; Error-code to global error variable
   ret
[end]


[_SetFileNameError]
$EXTERNAL _SetDOSError
   ;-------------------------------------------------------------------------------
   ; Set error code for Wild card error in file name
   ; Is called from File open routines if filename with wildcard is passed
   ;-------------------------------------------------------------------------------
_SetFileNameWildCardError0:
   clr r24                              ; file# to 0 for error (in Open file routines)
_SetFileNameWildCardError:
 * ldi r25 , cpFileNoWildCardAllowed
   rjmp _SetDOSError
[End]


[_ClearDOSError]
$EXTERNAL _SetDOSError
   ;-------------------------------------------------------------------------------
   ; Clear DOSError (write 0 to gbDOSError) and clears Carry-Flag
   ;-------------------------------------------------------------------------------
_ClearDOSError_add2toY:
   adiw yl, 2                           ; adjust Y-Pointer too if necessary
_ClearDOSError:
   clc
 * ldi r25 , cpNoError
   rjmp _SetDOSError1                   ; write cpNoError (0) to gbDOSError
[end]


[_ClearErrorFlag]
   ;-------------------------------------------------------------------------------
   ; Clear Error flag (=Carry-Flag) and adjust Y-pointer atcalling by ..._add2toY
   ;-------------------------------------------------------------------------------
_ClearErrorFlag_add2toY:
   adiw yl, 2
_ClearErrorFlag:
   clc
   ret
[end]


[_SetErrorFlag]
   ;-------------------------------------------------------------------------------
   ; Set Error flag (=Carry-Flag) and adjust Y-pointer atcalling by ..._add2toY
   ;-------------------------------------------------------------------------------
_SetErrorFlag_add2toY:
   adiw yl, 2                           ; adjust Y-pointer too if necessary
_SetErrorFlag:
   sec
   ret
[end]
```

```
[_SetBitFileStatusYZ]
   ;---------------------------------------------------------------------------
   ; Set Bit(s) in FileBufferStatus of current file-handle (using Z-Pointer)
   ; Input: Bits set in r24 are set in FileBufferStatus
   ;---------------------------------------------------------------------------
_SetBitFileEmptyYZ:
 * ldi r24 ,(dmFileEmpty + cp_FileBufferInitStatus)
   rjmp _SetBitFileStatusYZ
_SetBitEOFYZ:
 * ldi r24 ,(dmEOFinSector + cp_FileBufferInitStatus)       ; set file status to EOF in Sector
_SetBitFileStatusYZ:
   ldd zl, Y+0
   ldd zh, Y+1
   push r25
 * ldd r25 , Z + co_FileBufferStatus
   or r24, r25
 * std Z + co_FileBufferStatus , r24
   pop r25
   ret
[end]


[_ClearBitFileStatusYZ]
   ;---------------------------------------------------------------------------
   ; Clear Bit(s) in FileBufferStatus of current file-handle (using Z-Pointer)
   ; Input: Bits set in r24 are cleared in FileBufferStatus
   ;---------------------------------------------------------------------------
_ClearBitFileStatusYZ:
   ldd zl, Y+0
   ldd zh, Y+1
   push r25
 * ldd r25 , Z + co_FileBufferStatus
   com r24
   and r24, r25
 * std Z + co_FileBufferStatus , r24
   pop r25
   ret
[end]



[_SetFileWritePending]
$EXTERNAL _SetBitFileStatusYZ
   ;---------------------------------------------------------------------------
   ; Set the file write pending bit in the file buffer status byte
   ; Input Setting: File handle address in Y+0/1
   ;---------------------------------------------------------------------------
_SetFileWritePending:
   ; Input: Y0-> Filehandle
 * ldi r24 , dmWritePending
   rjmp _SetBitFileStatusYZ
[end]


[_SetDirFATWritePending]
   ;---------------------------------------------------------------------------
   ; set the DIR / FAT Write pending bit in the DIR / FAT buffer status byte
   ; Input-settings: none
   ; Entries for
   ; FAT: _SetFATWritePending
   ; DIR: _SetDIRWritePending
   ;---------------------------------------------------------------------------
*#IF cSepFATHandle = 1
_SetFATWritePending:
   lds r24, {gbFATBufferStatus}
 * sbr r24 , dmWritePending
   sts {gbFATBufferStatus}, r24
   ret
*#ELSE
```

```
_SetFATWritePending:
*#ENDIF
_SetDirWritePending:
    lds r24, {gbDirBufferStatus}
 * sbr r24 , dmWritePending
    sts {gbDirBufferStatus}, r24
    ret
[end]
```

```
   ;*** S E C T O R   R e a d i n g ********************************************

[_DriveReadSector_SEV]
$EXTERNAL _ClearDOSError , _SetDOSError
$EXTERNAL _DriveReadSector
   ;---------------------------------------------------------------------------
   ; Do low level Sector reading and sets DOSError variable depending on the result
   ; Driver Sector Read routine iscalled from here
   ; Input: X: Pointer to Sectornumber (LONG)
   ;         Z: SRAM-Address, to which Data transfer starts
   ;---------------------------------------------------------------------------
_DriveReadSector_SEV:
  call _DriveReadSector
   brcs _DriveReadSector_SEV1            ; Carry-Bit is error-flag
   rjmp _ClearDOSError
_DriveReadSector_SEV1:
   rjmp _SetDOSError
[End]


[_DriveWriteSector_SEV]
$EXTERNAL _ClearDOSError , _SetDOSError
$EXTERNAL _DriveWriteSector
   ;---------------------------------------------------------------------------
   ; Do low level Sector Writing and sets DOSError variable depending on the result
   ; Driver Sector Write routine iscalled from here
   ; Input: X: Pointer to Sectornumber (LONG)
   ;         Z: SRAM-Address, from which Data transfer starts
   ;---------------------------------------------------------------------------
_DriveWriteSector_SEV:
  call _DriveWriteSector
   brcs _DriveWriteSector_SEV1           ; Carry-Bit
   rjmp _ClearDOSError
_DriveWriteSector_SEV1:
   rjmp _SetDOSError
[End]


[_ReadDirSector]
$EXTERNAL _DriveReadSector_SEV
   ;---------------------------------------------------------------------------
   ; This routine provides a sector buffer (Normally DIR) to the _GetFileSystem routine
   ; Can used only, if file system is not in use!!
   ; Input Settings none
   ; Result: Sector from Drive is loaded to SRAM starting at gbDIRBuffer(1)
   ;         driver-Error is set in gbDOSError
   ;---------------------------------------------------------------------------
_ReadDirSector:
   loadadr glDirSectorNumber , X
   loadadr gbDirBuffer(1) , Z
   rjmp _DriveReadSector_SEV
[end]
```

```
[_GetSector]
$EXTERNAL _A2LongX , _AddrFileSectorNumber2X , _CleanHWStack , _ClearErrorFlag , _DriveReadSector_SEV
$EXTERNAL _DriveWriteSector_SEV , _LongX2B , _SaveSector2FATCopy
$EXTERNAL _COMP32
   ;---------------------------------------------------------------------------
   ; get a Sector from the Drive. It loads file sector, FAT Sector or DIR sector
   ; If desired sector is same as just loaded sector, than do nothing
   ; if just loaded sector was changed (Bit WritePending set) then save current loaded
   ; sector before loade new one.
   ; For FAT sector, copy is also written, in necessary
   ; Input: Sector# in A
   ;         File handle address in Y+0/1 for file sector
   ; Entries: _GetSector_FAT for FAT sectors
   ;          _GetSector_Dir for DIR sectors
   ;          _GetSector_File for file sectors
   ; Output: Sector loaded
   ;          Z points to Sector buffer
   ;          If new sector loaded, than T-Flag set, otherwise cleared
   ;---------------------------------------------------------------------------
_GetSector_FAT:
 * ldi _temp1 , cp_FATBufferInitStatus
*#IF cSepFATHandle = 1
   loadadr glFATSectorNumber , X
*#ELSE
   loadadr glDirSectorNumber , X
*#ENDIF
   rjmp _GetSector
_GetSector_Dir:
 * ldi _Temp1 , cp_DirBufferInitStatus
   loadadr glDirSectorNumber , X
   rjmp _GetSector
_GetSector_File:
   rcall _AddrFileSectorNumber2X         ; poiner of sector# to X
 * ldi _Temp1 , cp_FileBufferInitStatus
_GetSector:
   rcall _LongX2B                        ; load number of current sector to B
  call _Comp32
   brne _GetSector1
   ; Sector is yet loaded
   ; restore Pointers and return
   clt                                   ; T-Flag cleared for <no new sector loaded>
_GetSector91:
   adiw xl, 1                            ; bring X to buffer-Start
   ;_GetSector92:
   mov zl, xl                            ; Bufferstart to Z
   mov zh, xh
   sbiw xl, 5                            ; Restore Pointer to sector-number
   rjmp _ClearErrorFlag

_GetSector1:
   ;---------------------------------------------------------------------------
   ;   ld _temp2, x+                         ; load Buffer-Status
   ;   ; X point now to Bufferstart
   ;   ; preserve bit: dFATDirSaveAtEnd + dFATDirSaveAnyWay to new bufferstatus
   ;   mov zl, _temp2
   ; * andi zl ,(dmFATDirSaveAtEnd + dmFATDirSaveAnyWay + dmFATDirWritePending)
   ;   or _temp1, zl                         ; copy bits to new status
   ;   mov zl, xl                            ; Bufferstart to Z
   ;   mov zh, xh
   ;   st -x, _temp1                         ; Save new Buffer-Status
   ;---------------------------------------------------------------------------
   ld _temp2, X+                         ; current Status
   mov zl, xl                            ; Bufferstart to Z
   mov zh, xh
   mov r23, _temp2
 * cbr r23, ( dmWritePending + dmFATSector )        ; clear write pending and old FAT-Status
   or _temp1, r23                        ; combine Init status with relevant bits of old status
   st -X, _temp1                         ; save new buffer-status
   ;---------------------------------------------------------------------------
   sbiw xl, 4                            ; Restore Pointer to sector-number
 * sbrs _temp2 , dWritePending           ; Must sector written be back
   rjmp _GetSector4
   ; Save Sector
   push _A1                              ; save new sector number to stack
```

```
   push _A2
   push _A3
   push _A4
   push _temp2                          ; old buffer status
   push xl                              ; save pointer to sectornumber to stack
   push xh
   rcall _DriveWriteSector_SEV
   pop xh
   pop xl
   pop _temp2                           ; old buffer status
   brcs _GetSector5                     ; Error in low level Drive-routine
  * sbrs _temp2 , dFATSector            ; Is old sector a FAT Sector
   rjmp _GetSector3
   push xl                              ; save pointer to sectornumber to stack
   push xh
   rcall _SaveSector2FATCopy
   pop xh
   pop xl
   brcs _GetSector5                     ; error in drive routine
_GetSector3:
   pop _A4
   pop _A3
   pop _A2
   pop _A1
_GetSector4:
   rcall _A2LongX
   rcall _GetSector91
   push zl
   push zh
   rcall _DriveReadSector_SEV           ; Get new sector
   pop zh                               ; Restore pointer to buffer
   pop zl
   set                                  ; T -Flag set for < New sector loaded>
   ret
_GetSector5:
   rjmp _CleanHWStack4
[end]


[_LoadNextFileSector]
$EXTERNAL _Add32_Word , _AddrFileSectorNumber2X , _CheckAvailableSectorSize , _GetNextSectorNumber
$EXTERNAL _GetSector , _LoadFileBufferStatusYZ , _LoadFilePositionYZ2A , _LoadFileSectorNumberYZ2A
$EXTERNAL _SaveFilePositionA2YZ , _SetBitFileStatusYZ , _SetDOSError
   ;---------------------------------------------------------------------------
   ; Load next sector of a file
   ; Input: file handle address a Y+0/1
   ;         Entry: _LoadNextFileSector: File position pointer must be just in new sector
   ;                _LoadNextFileSector_Position: routine adjust file pointer to next loaded
   ;                                              sector
   ;---------------------------------------------------------------------------
_LoadNextFileSector_Position:
   ; with adjust of File Position to start of next sector
   rcall _LoadFileBufferStatusYZ
 * andi r24 , dmEOFAll                  ; generell EOF or in(last) Sector
   brne _LoadNextFileSector82           ; EOF
   rcall _LoadFilePositionYZ2A          ; load File position to A
   clr r16                              ; bring position to base of sector
   andi r17, &B11111110
   clr r20
   ldi r21, &H02                        ; add 512
   rcall _Add32_Word
   rcall _SaveFilePositionA2YZ
_LoadNextFileSector:
   rcall _LoadFileBufferStatusYZ
 * andi r24 , dmEOFAll                  ; generell EOF or in(last) Sector
   brne _LoadNextFileSector82           ; EOF
   rcall _LoadFileSectorNumberYZ2A      ; sector# to A
   rcall _GetNextSectorNumber
   brcs _LoadNextFileSector81
   rcall _AddrFileSectorNumber2X        ; address of Sector-number to X
   rcall _getSector_File
   brcs _LoadNextFileSector83           ; Error in drive routine?
   rjmp _CheckAvailableSectorSize
```

```
_LoadNextFileSector81:
 * cpi r25 , cpNoNextCluster             ; set EOF at this Error too
   brne _LoadNextFileSector83
   rcall _SetBitEOFYZ
 * ldi r25 , cpNoNextCluster
   rjmp _SetDOSError
_LoadNextFileSector82:
 * ldi r25 , cpEndOfFile
_LoadNextFileSector83:
   rjmp _SetDOSError
[end]


   ;*** D I R E C T O R Y   H a n d l i n g ************************************


[_SearchDirEntry]
$EXTERNAL _B2WordX , _ClearDOSError , _Comp_String_Mem , _GetDirEntry , _GetNextDirSectorNumber
$EXTERNAL _GetSector , _NormFileName , _SetDOSError , _SetErrorFlag , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; Search DirEntry to a Filename+Ext.
   ; Input:  norm. Filename+Ext: Pointer in Z => _SearchDirEntry
   ;          Filename+ext: pointer in X => _SearchDirEntry_CheckFileName
   ; Output: Found: C-Flag cleared
   ;         DirEntry#: r20-r21, gwDirEntry
   ;         Sector-Number: A (r16-r19)
   ;         DirEntry: Z-Pointer to SRAM
   ; Entry _SearchDirEntry_CheckFileName normalizes Filename and checked for wildcard
   ;       too.
   ;---------------------------------------------------------------------------
   ; Alternate Entry with checking of Filename of no Wildcard
_SearchDirEntry_CheckFileName:
   rcall _NormFileName
   breq _SearchDirEntry                  ; no Wildcard?
   rjmp _SetFileNameWildcardError        ; set DOS Error
   ; Normal Entry of Routine
_SearchDirEntry:
   st -Y, zh
   st -Y, zl
   rcall _GetDirEntry_DirStart
   ; Z points now to buffer
_SearchDirEntry2:
   brcc _SearchDirEntry2b
   rjmp _SetErrorFlag_add2toY
_SearchDirEntry2b:
   ldd r24, Z+11                         ; check for long file name entry
   cpi r24, &H0F                         ; LFN have 0F in attribut-byte
   breq _SearchDirEntry3
 * cbr r24, cpFileSearchAllowed
   brne _searchDirEntry3
   ld r24, Z                             ; load first char of name
   cpi r24, ASC(".")                     ; dont show special Dir Entries . and  ..
   breq _SearchDirEntry3
   cpi r24, &HE5                         ; deleted?
   breq _SearchDirEntry3
   cpi r24, &H00                         ; Directory End?
   breq _SearchDirEntry4
   ldd xl, y+0                           ; Pointer to Filename+Extend
   ldd xh, y+1
   ld r25, X
   cpi r25, asc("?")
   breq _SearchDirEntry2a
   cpse r25, r24
   rjmp _SearchDirEntry3                 ; check here first char for faster search
_SearchDirEntry2a:
   ldi r23, 11                           ; Compare 11 Characters
   rcall _Comp_String_Mem
   breq _SearchDirEntry5                 ; Found? goto finish
_SearchDirEntry3:
   adiw zl, 32                           ; Z-Pointer to next Dir Entry
   ldi r24, 1
   add r20, r24
```

```
   brhc _SearchDirEntry2                ; Sector overflow = 16 Entries = H-Carry-Bit
   clr r24
   adc r21, r24                         ; Complete Root-Entry Incr
   push r21                             ; save DirEntry#
   push r20
   rcall _GetNextDirSectorNumber
   brcs _SearchDirEntry6
   rcall _GetSector_Dir
_SearchDirEntry6:
   pop r20
   pop r21
   brcc _SearchDirEntry2
  * cpi r25 , cpDriverErrorStart
   brcc _SearchDirEntry4a
_SearchDirEntry4:
  * ldi r25 , cpFileNotFound
_SearchDirEntry4a:
   ld zl, Y+                            ; pointer to filename back to Z
   ld zh, Y+                            ; and restore Soft stack pointer
   rjmp _SetDOSError
_SearchDirEntry5:
   loadadr gwDirEntry , X
   rcall _B2WordX
   rjmp _ClearDOSError_add2toY
[end]


[_GetNextDirSectorNumber]
$EXTERNAL _Add32_Word , _CleanHWStack , _ClearDOSError , _GetNextSectorNumber , _LongX2A , _LongX2B
$EXTERNAL _SetDOSError
$EXTERNAL _COMP32
   ;---------------------------------------------------------------------------
   ; get next sectornumber to a directory
   ; check Root Entry overflow of FAT16 Root
   ; and end of cluster chain in other directories
   ; input DirEntry# in B (will be destroyed)
   ; Directory buffer must be actually handled (glDirSectorNumber must really contain a DIR-Sector#)
   ;---------------------------------------------------------------------------
_GetNextDirSectorNumber:
   lds r0, {gbFileSystemStatus}         ; Check for Root Directory at FAT16
 * sbrc r0, dFileSystemStatusFAT
   rjmp _GetNextDirSectorNumber2        ; jump if FAT32
   loadadr glDirFirstSectorNumber , X
   rcall _LongX2A
   loadadr glRootFirstSector , X
   push r21
   push r20
   rcall _LongX2B
   call _Comp32
   pop r20
   pop r21
   brne _GetNextDirSectorNumber2        ; Branch if Sub-Dir in FAT16
   loadadr gwRootEntries , X            ; check last FAT16 Root -Entry reached
   ld r22, x+
   ld r23, x+
   cp r20, r22
   cpc r21, r23
   brcs _GetNextDirSectorNumber1        ; Carry on if within Root Directory
  * ldi r25 , cpNoFreeDirEntry          ; End of FAT16 Root Directory
   rjmp _SetDOSError
_GetNextDirSectorNumber1:
   rcall _GetNextDirSectorNumber80      ; root in FAT16
   rcall _Add32_1
   rjmp _ClearDOSError
_GetNextDirSectorNumber2:
   ; FAT16 Sub_Directory and FAT32 general
   rcall _GetNextDirSectorNumber80
   push r19                             ; save sector# for no success
   push r18
   push r17
   push r16
   rcall _GetNextSectorNumber
   brcs _GetNextDirSectorNumber3
```

```
   rjmp _CleanHWStack4
_GetNextDirSectorNumber3:
   pop r16                          ; restore old sector# in case of error
   pop r17
   pop r18
   pop r19
   ret
_GetNextDirSectorNumber80:
   loadadr glDirSectorNumber , X    ; Current Sector Number
   rjmp _LongX2A
[end]



[_GetFreeDirEntry]
$EXTERNAL _A2LongX , _B2WordX , _CleanHWStack , _ClearDOSError , _ClearFileSector , _ClusterA2FAT
$EXTERNAL _GetDirEntry , _GetFreeCluster , _GetNextCluster , _GetNextDirSectorNumber , _GetSector
$EXTERNAL _LastCluster2FAT , _LongX2A , _Mov4A2B , _Sector2Cluster , _SetDOSError
   ;-------------------------------------------------------------------------------
   ; get a free DIR Entry from directory area
   ; Start searching at following entries:
   ; o _GetFreeDirEntry: DirENtry# in gwFreeDirEntry (last free found Dir Entry)
   ; Result: DirEntry# in B (r20,r21)
   ;-------------------------------------------------------------------------------
_GetFreeDirEntry:
   rcall _GetDirEntryFreeStart
   ; Z points now to buffer
   brcs _GetFreeDirEntry6           ; Error in former (drive) routine
_GetFreeDirEntry2:
   ld r24, Z                        ; load first char of name
   cpi r24, &HE5                    ; deleted?
   breq _GetFreeDirEntry5
   cpi r24, &H00                    ; Directory End?
   breq _GetFreeDirEntry5
   adiw zl, 32                      ; Z-Pointer to next Dir Entry
   ldi r24, 1
   add r20, r24
   brhc _GetFreeDirEntry2           ; Sector overflow = 16 Entries = H-Carry-Bit
   clr r24
   adc r21, r24                     ; Complete Dir-Entry Incr
   push r21
   push r20
   rcall _GetNextDirSectorNumber
   brcs _GetFreeDirEntry4
   rcall _GetSector_DIR
_GetFreeDirEntry4:
   pop r20
   pop r21
   brcc _GetFreeDirEntry2
 * cpi r25 , cpNoNextCluster
   brne _GetFreeDirEntry6
   push r21
   push r20
   rcall _Sector2Cluster            ; cluster# of current Dir-cluster
   push r19
   push r18
   push r17
   push r16
   rcall _GetFreeCluster
   brcs _GetFreeDirEntry7
   rcall _LastCluster2FAT           ; EOC Mark to new cluser
   rcall _Mov4A2B                   ; new cluser# to B (temporary save)
   pop r16                          ; previous cluster#
   pop r17
   pop r18
   pop r19
   push r23                         ; new cluste# to stack
   push r22
   push r21
   push r20
   rcall _GetNextCluster
   pop r16                          ; new cluster#
   pop r17
```

```
   pop r18
   pop r19
   brcc _GetFreeDirEntry8a
 * cpi r25, cpNoNextCluster
   brne _GetFreeDirEntry8
   rcall _ClusterA2FAT
_GetFreeDirEntry4a:
   rcall _ClearDirCluster
   rjmp _GetFreeDirEntry4
_GetFreeDirEntry5:
   loadadr gwDirEntry , X
   rcall _B2WordX                       ; Save DirEntry#
   rcall _LongX2A                       ; DirSectorNumber# to A
   loadAdr gwFreeDirEntry , X
   rcall _B2WordX                       ; save last free found DirEntry#
   rcall _A2LongX                       ; and DirSector#
   rjmp _ClearDOSError
_GetFreeDirEntry6:
   rjmp _SetDOSError
_GetFreeDirEntry7:
   rjmp _CleanHWStack6
_GetFreeDirEntry8a:
 * ldi r25, cpClusterError
   rcall _SetDOSError
_GetFreeDirEntry8:
   rjmp _CleanHWStack2
[end]



[_GetDirEntry]
$EXTERNAL _ClearDOSError , _GetSector , _LongX2A , _SetErrorFlag , _ShiftL_WordB , _WordX2B
   ;-------------------------------------------------------------------------------
   ; Get Pointer to Dir Entry in Dir-Sector (load if neccessary)
   ; Input:  DirEntry#: B
   ; Output: Z points to DirEntry
   ;         DirEntry# unchanged in r20/r21
   ;-------------------------------------------------------------------------------
_GetDirEntry_Y:
   ldd xl, Y+0
   ldd xh, Y+1
 * adiw xl , co_FileDirEntry
   rjmp _GetDirEntry1
_GetDirEntry_DirStart:
   clr r20
   clr r21
   loadadr glDirFirstSectorNumber , X
   rjmp _GetDirEntry2
_GetDirEntry_0:
   loadadr gwDir0Entry , X
   rjmp _GetDirEntry1
_GetDirEntryFreeStart:
   loadadr gwFreeDirEntry , X
_GetDirEntry1:
   rcall _WordX2B
_GetDirEntry2:
   rcall _LongX2A
_GetDirEntry:
   st -Y, r21
   st -Y, r20                           ; save of DirEntry#
   rcall _GetSector_Dir                 ; check/load sector; Z-> Sector-Buffer
   brcs _GetDirEntry9
   ldd r20, Y+0                         ; LSB of DirEntry#
   andi r20, &H0F                       ; build sector-Offset
   clr r21
   ldi r24, 5
   rcall _ShiftL_WordB
   add zl, r20                          ; Z just at base of sector-buffer
   adc zh, r21
   ld r20, Y+
   ld r21, Y+
   rjmp _ClearDOSError
_GetDirEntry9:
```

```
   rjmp _setErrorFlag_Add2toY
[end]


[_SearchNextDirEntry]
$EXTERNAL _GetDirEntry , _SearchDirEntry
   ;---------------------------------------------------------------------------
   ; Search next Dir-Entry to Filename+Ext behind a given DirEntry#
   ; Output: See _SearchDirEntry
   ;---------------------------------------------------------------------------
_SearchNextDirEntry:
   loadadr gwDir0Entry , X
   rcall _GetDirEntry1
   push zh
   push zl
   loadadr gsDir0TempFileName , Z
   st -Y, zh
   st -Y, zl
   pop zl
   pop zh
   rjmp _SearchDirEntry3                ; goto next Dir entry and search
[end]


[_Comp_String_Mem]
   ;---------------------------------------------------------------------------
   ; compares two string in Mem for Search Directory and consider wildcard "?"
   ; Input:  first string -> Z-Pointer
   ;         second String -> X-Pointer (String must End with &H00)
   ;         max length of String: r23
   ; Output: Found: Z-Flag set
   ;---------------------------------------------------------------------------
_Comp_String_Mem:
   push zh
   push zl
_Comp_String_Mem1:
   ld r24, z+
   ld r25, x+
   cpi r25, asc("?")
   breq _Comp_String_Mem3
   cp r24, r25
   brne _Comp_String_Mem2
_Comp_String_Mem3:
   dec r23
   brne _Comp_String_Mem1
_Comp_String_Mem2:
   pop zl
   pop zh
   ret
[end]


[_NormFileName]
$EXTERNAL _UPPER
   ;---------------------------------------------------------------------------
   ; Normalize Filename for Search in Directory
   ; Input:  X-> Source-string
   ;             Output-string to gsTempFileName with entrance _NormFileName
   ;                          to ->X with entrance _NormFileName0
   ; Output: Z-> Target-String
   ;         r25: count of "?", for wildcard check, if no wildcard Z=1
   ;---------------------------------------------------------------------------
_NormFileName:
   loadadr gsTempFileName , Z           ; output of routine to gsTempFileName
_NormFileName0:
   push r24
   clt
   clr r25                              ; Clear flag for repeating " " of ?
   ldi r23, 11                          ; 11 characters to check
_NormFileName1:
   brts _NormFileName5                  ; No new char to read in?
```

```
   ld r24, X+                          ; read char in
   cpi r24, asc("*")                   ;
   brne _NormFileName3
   ldi r24, asc("?")
   Set
   rjmp _NormFileName5
_NormFileName3:
   cpi r24, asc(".")
   brne _NormfileName4
   cpi r23, 3                          ; just 8 char handled?, on more action for point
   brne  _Normfilename3A
   clt
   rjmp _NormFileName1
_NormFileName3a:
   ldi r24, 32                         ; blank
   Set
   rjmp _NormFileName5
_NormFileName4:
   cpi r24, 0
   brne _NormfileName4a
   ldi r24, 32
   rjmp _NormFileName7
_NormFileName4a:
  call _Upper
_NormFileName5:
   st Z+, r24
   cpi r24, asc("?")
   brne _NormFileName5a
   inc r25
_NormFileName5a:
   cpi r23, 4
   brne _NormFileName6
   clt
_NormFileName6:
   dec r23
   brne _NormfileName1
   rjmp _NormfileName9
_NormFileName7:
   ldi r24, 32                         ; Fill to End of Name with blank
_NormFileName8:
   st Z+, r24
   dec r23
   brne _NormFileName8
_NormFileName9:
   ldi r24, 0
   st Z+, r24
   sbiw zl, 12
   tst r25                             ; if no wildcard return Z-Flag
   pop r24
   ret
[end]


[_GetFileName]
$EXTERNAL _Mem_Copy
   ;---------------------------------------------------------------------------
   ; Copy Filename from DirEntry to String
   ; Input:  DirEntry: Z-Pointer
   ;         target-string: X-Pointer (with at least 8 Char)
   ;---------------------------------------------------------------------------
_GetFileName:
   ldi r24, 8
   rcall _Mem_Copy
   sbiw zl, 8
   clr r24
   st X+, r24                          ; String Terminator
   ret
[end]



[_GetFileExtend]
$EXTERNAL _Mem_Copy
   ;---------------------------------------------------------------------------
```

```
   ; copy the Extend of the file name from DIR-Buffer to a string
   ; Input: Z points to SRAM-Directory Entry
   ; X Points to Out-Put-String with at least 3 char
   ;-----------------------------------------------------------------------------
_GetFileExtend:
   adiw zl, 8
   ldi r24, 3
   rcall _Mem_Copy
   sbiw zl, 11
   clr r24
   st X+, r24                          ; String Terminator
   ret
[end]


[_GetFileAttribute]
   ;-----------------------------------------------------------------------------
   ; copy the attribut-byte of a file from the DIR-buffer to a SRAM-Byte
   ; Input: Z points to SRAM-Directory Entry
   ; X Points to Byte to get FileAttribute
   ;-----------------------------------------------------------------------------
_GetFileAttribute:
   ldd r24, Z+11
   st X+, r24
   ret
[end]


[_GetFileFirstCluster]
   ;-----------------------------------------------------------------------------
   ; Copy first cluster# of a file from the DIR-Buffer to a SRAM-Word
   ; Input: Z points to SRAM-Directory Entry
   ; X Points to Word to Hold First Cluster Number
   ;-----------------------------------------------------------------------------
_GetFileFirstCluster:
   ldd r25, Z+26
   st X+, r25
   ldd r25, Z+27
   st X+, r25
   lds r0, {gbFileSystemStatus}
 * sbrs r0 , dFileSystemStatusFAT
   rjmp _GetFileFirstCluster1
   ldd r25, Z+20
   st X+, r25
   ldd r25, Z+21
   st X+, r25
   ret
_GetFileFirstCluster1:
   clr r25
   st X+, r25
   st X+, r25
   ret
[end]


[_GetFileDate]
   ;-----------------------------------------------------------------------------
   ; get the File Date of a file from the DIR-Buffer, convert it to Day/Month/Year
   ; and write it to a 3-Byte array in SRAM
   ; Input: Z points to SRAM-Directory Entry
   ; X Points to Byte(s) Day, Month, Year
   ;-----------------------------------------------------------------------------
_GetFileDate:
   ldd r22, z+24
   ldd r23, z+25
   mov r24, r22
   andi r24, &B00011111
   st x+, r24
   lsr r23
   ror r22
   swap r22
```

```
   andi r22, &H0F
   st x+, r22
   subi r23, 20
   brpl  _GetFileDate1
   subi r23, 156
_GetFileDate1:
   st x+, r23
   ret
[end]


[_GetFileTime]
   ;---------------------------------------------------------------------------
   ; get the File time of a file from the DIR-Buffer, convert it to Sec/Min/Day
   ; and write it to a 3-Byte array in SRAM
   ; Input: Z points to SRAM-Directory Entry
   ; X Points to Byte(s) Second, Minute, Hour
   ;---------------------------------------------------------------------------
_GetFileTime:
   ldd r22, z+22
   ldd r23, z+23
   mov r25, r22
   andi r25, &B00011111
   lsl r25                              ; Seconds
   st x+, r25
   lsr r23
   ror r22
   lsr r23
   ror r22
   lsr r23
   ror r22
   lsr r22
   lsr r22
   st x+, r22
   st x+, r23
   ret
[end]


[_GetFileSize]
$EXTERNAL _Mem_Copy
   ;---------------------------------------------------------------------------
   ; Copy the file size of a file from the DIR-Buffer to a SRAM-LONG
   ; Input: Z points to SRAM-Directory Entry
   ; X Points to Long Variable
   ;---------------------------------------------------------------------------
_GetFileSize:
   adiw zl, 28
   rcall _Mem_Copy_4
   sbiw zl, 32
   ret
[end]


   ;*** F I L E   H a n d l i n g ************************************************

[_SetFileHandle]
$EXTERNAL _ClearDOSError , _ClearFileHandle , _CompareRange , _Mem_Copy , _SetDOSError
   ;---------------------------------------------------------------------------
   ; Input: r24: FileNumber
   ;        r20, r21: Root-Entry-Number
   ;        r25: File-Open-Mode
   ; Output:
   ;        OK: Pointer to file-handle block in Z; Carry-Flag Cleared
   ;        Not OK: Carry-Flag Set; Error-Code in gbDOSError and r25
   ;            for: File-number in Use, File just open in write-mode
   ;---------------------------------------------------------------------------
_SetFileHandle:
   tst r24                              ; is file# = 0
   brne _SetFileHandle1a
 * ldi r25 , cpFileNumberInvalid
```

```
   rjmp _SetDOSError
_SetFileHandle1a:
 * ldi r19 , cFileHandles
   loadadr abFileHandles(1) , Z
   clt                                  ; T-Flag = found free handle
_SetFileHandle1:
   ld r18, Z                            ; get file-number of block
   tst r18                              ; free handle?
   brne  _SETFILEHANDLE2
   brts _SetFileHandle3                 ; free handle yet saved
   mov r14, zl                          ; save pointer of free block
   mov r15, zh
   set                                  ; sign for free handle found
   rjmp _SetFileHandle3                 ; check next block
_SetFileHandle2:
   cp r18, r24                          ; File-number in use?
   breq _SetFileHandle5                 ; Yes, goto Error
   push zl                              ; Check new file is just loaded in a
   push zh                              ; filehandle
   push r24
 * adiw zl , co_FileDirEntry
   loadadr gwDirEntry , X               ; check DirEntry and Sector#
   rcall _CompareRange
   pop r24
   pop zh
   pop zl
   brne _SetFileHandle3                 ; Not in use
 * ldd r17 , Z + co_FileMode            ; check for kind of use(read or write)
   and r17, r25                         ; only valid if both set read (only)
 * sbrs r17 , dReadOnly                 ; one of both is writePrint
   rjmp _SetFileHandle6                 ; Open-Error
_SetFileHandle3:
   dec r19                              ; handle-counter
   breq _SetFileHandle4                 ; all handle searched
 * subi zl , lByte(c_FileHandleSize_M)  ; advance filehandle pointer to next block
 * sbci zh , hbyte(c_FileHandleSize_M)
   rjmp _SetFileHandle1
_SetFileHandle4:
   brtc _SetFileHandle7
   mov zl, r14
   mov zh, r15
   ; Clear File Handle
   rcall _ClearFileHandle
   std z+0, r24                         ; set file-number
 * std z + co_FileMode , r25            ; Open mode
   push zl                              ; Check new file is just loaded in a
   push zh                              ; filehandle
   push r24                             ; save file#, r25 is saved in _mem_Copy
   ldi r24, 6                           ; check dir Entry and Dir Sector
 * adiw zl , co_FileDirEntry
   mov xl, zl
   mov xh, zh
   loadadr gwDirEntry , Z               ; Copy DirEntry and DirSector# to File handle
   ldi r24, 6
   rcall _Mem_Copy
   pop r24
   pop zh
   pop zl
   rjmp _ClearDOSError
_SetFileHandle5:
 * ldi r25 , cpFileOpenHandleInUse
   rjmp _SetDOSError
_SetFileHandle6:
 * ldi r25 , cpFileOpenShareConflict
   rjmp _SetDOSError
_SetFileHandle7:
 * ldi r25 , cpFileOpenNoHandle
   rjmp _SetDOSError
[end]
```

```
[_ClearFileHandle]
$EXTERNAL _AddrFileHandleBase2Z
_ClearFileHandle_Y:
   rcall _AddrFileHandleBase2Z
_ClearFileHandle:
 * ldi r16 , co_FileBuffer
   clr r0
_ClearFileHandle1:
   st Z+, r0
   dec r16
   brne _ClearFileHandle1
 * sbiw zl , co_FileBuffer               ; restore file handle base address
   ret
[end]


[_GetFileHandle]
$EXTERNAL _SetDOSError
   ;---------------------------------------------------------------------------
   ; Get the Base address of File-handle to a file number
   ; Input:  File number in r24
   ; Output: File handle -> Z-Pointer
   ;         Not Found: C-Flag Set, DOS Error set
   ;---------------------------------------------------------------------------
_GetFileHandle:
   tst r24                               ; Zero in file# not allowed
   breq _GetFileHandle3                  ; if File# = 0 then Error!
 * ldi r23 , cFileHandles                ; max. Number of file -handles
   loadadr abFileHandles(1) , Z
_GetFileHandle1:
   ld r25, Z
   cp r25, r24
   breq _GetFileHandle5
   dec r23
   breq _GetFileHandle2                  ; All filehandle searched
 * subi zl , lByte(c_FileHandleSize_M)   ; advance filehandle pointer to next block
 * sbci zh , hbyte(c_FileHandleSize_M)
   rjmp _GetFileHandle1
_GetFileHandle2:
 * ldi r25 , cpFileNumberNotFound
   rjmp _SetDOSError
_GetFileHandle3:
 * ldi r25 , cpFileNumberInvalid
   rjmp _SetDOSError
_GetFileHandle5:
   ret
[end]


[_GetFreeFileNumber]
$EXTERNAL _ClearDOSError , _GetFileHandle , _SetDOSError
   ;---------------------------------------------------------------------------
   ; Search a free file# starting at &H80
   ; Return file# in r24
   ;---------------------------------------------------------------------------
_GetFreeFileNumber:
   ldi r24, &H80                         ; use range from 8O to FF for FS defined File#
_GetFreeFileNumber1:
   rcall _GetFileHandle
   brcs _GetFreeFileNumber2
   inc r24
   brne _GetFreeFileNumber1
 * ldi r25 , cpNoFreeFileNumber
   clr r24                               ; set file# to 0 in case of error
   rjmp _SetDOSError
_GetFreeFileNumber2:
   rjmp _ClearDOSError
[end]
```

```
[_CloseFileHandle]
$EXTERNAL _AddrFileHandleBase2Z , _ClearFileHandle , _GetFileHandle , _SaveFileSector
$EXTERNAL _SetBitFileStatusYZ
   ;---------------------------------------------------------------------------
   ; Close filehandle and save current sector to Disk if neccessary
   ; Input: different entries:
   ;         _CloseFileHandle_Y: File handle address in Y+0/1
   ;         _CloseFileHandle: file# in r24
   ;---------------------------------------------------------------------------
_CloseFileHandle_Y:
   rcall _AddrFileHandleBase2Z
   rjmp _CloseFileHandle1
_CloseFileHandle:
   ; File number in r24
   rcall _GetFileHandle
   brcs _CloseFileHandle2
_CloseFileHandle1:
 * ldi r24 , dmFATDirSaveAnyWay
   rcall _SetBitFileStatusYZ
   rcall _SaveFileSector                ; clear file unconditional of error
   rjmp _ClearFileHandle_Y
_CloseFileHandle2:
   ret
[end]


[_OpenFile]
$EXTERNAL _AddrFileFirstCluster2X , _AddrFileSize2X , _CheckA_Zero , _CheckAvailableSectorSize
$EXTERNAL _ClearErrorFlag , _Cluster2Sector , _GetFileFirstCluster , _GetFileSize , _GetSector
$EXTERNAL _LoadFileBufferStatusYZ , _LoadFileFirstClusterYZ2A , _SearchDirEntry , _SetBitFileStatusYZ
$EXTERNAL _SetDOSError , _SetErrorFlag , _SetFileHandle
   ;---------------------------------------------------------------------------
   ; Open a file
   ; Input: Z: Filename + extend (normalized)
   ;        r24: File-number
   ;        r25: File-open-mode
   ;---------------------------------------------------------------------------
_OpenFile:
   push r24                             ; Save file-number to stack
   push r25                             ; save open mode to stack
   rcall _SearchDirEntry
   pop r23                              ; don;t destroy Error Code
   pop r24
   brcs _OpenFile9                      ; Error occured
   mov r25, r23                         ; File open mode to r25
   ldd r22, z+11                        ; Get File Attribut
   eor r22, r25                         ; Check File-attr. read-only bit and Open-mode read/write
   sbrs r22, 0                          ; if bit0=0 than OK: read/read or write/write => 0
   rjmp _Openfile1
   eor r22, r25
   sbrc r22, 0                          ; if Bit0=0 than OK: open-mode = read than 1/1 => 0
   rjmp _OpenFile92                     ; try to open read-only file in write mode
_OpenFile1:
   push zh                              ; save Address to Dir entry
   push zl
   rcall _SetFileHandle
   st -Y, zh                            ;
   st -Y, zl
   pop zl                               ; Address of Dir -Entry
   pop zh
   brcs _OpenFile9                      ; first restore Stack before checking Error in _SetFileHandle
   rcall _AddrFileSize2X                ; load file size from Directory to file handle
   rcall _GetFileSize
   ; File position was set to 0 by _SetfileHandle
   rcall _AddrFileFirstCluster2X        ; load First Cluster# from directory to file handle
   rcall _GetFileFirstCluster           ;
   rcall _LoadFileFirstClusterYZ2A      ; first cluster# from Filehandle to A
   rcall _CheckA_Zero                   ; check File-Empty
   breq _OpenFile91                     ; File empty -> EOF
   rcall _Cluster2Sector                ; put number of first sector to A
   rcall _GetSector_File
   brcs _OpenFile93
   rcall _CheckAvailableSectorSize
```

```
_OpenFile2:
   rcall _LoadFileBufferStatusYZ         ; store to Z for convenience of next function
   rjmp _ClearErrorFlag_Add2toY
_Openfile9:
   ret
_OpenFile91:
   rcall _SetBitFileEmptyYZ
   rjmp _OpenFile2
_OpenFile92:
 * ldi r25 , cpFileReadOnly
   rjmp _SetDOSError
_OpenFile93:
   rjmp _SetErrorFlag_Add2toY
[end]



   ;*** F i l e   -   R e a d i n g ******************************************
[_FileReadByte]
$EXTERNAL _A2LongZ , _Add32_Word , _ClearDOSError , _LoadNextFileSector , _LongZ2A , _LongZ2B
$EXTERNAL _SetDOSError
$EXTERNAL _COMP32
   ;---------------------------------------------------------------------------
   ; Read one byte from a file
   ; Input:  File handle in Y0,+1
   ; Output: OK: read byte: r24
   ;         Not OK: C-Falg Set, Error Code in r25
   ;---------------------------------------------------------------------------
   ; in this routine most of file-handle is done in routine for speed optimizing
_FileReadByte:
   ldd zl, Y+0
   ldd zh, Y+1
 * ldd r25 , Z + co_FileMode
 * andi r25 , cFileRead_Mode
   breq _FileReadByte98                  ; not allowed File mode
_FileReadByte0:
   ldd zl, Y+0
   ldd zh, Y+1
 * ldd r24 , Z + co_FileBufferStatus     ; check yet EOF
 * sbrc r24 , dEOF
   rjmp _FileReadByte81                  ; yet EOF
 * adiw zl , co_FilePosition
   rcall _LongZ2A                        ; load File Position
   mov r25, r17                          ; sector position
   mov r24, r16
   rcall _Add32_1                        ; Advance file position
   sbiw zl, 4                            ; Z-Pointer back to file position
   rcall _A2LongZ                        ; save new File position
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileSize
   rcall _LongZ2B                        ; load file size
  call _Comp32                           ; EOF?
   brcs _FileReadByte1                   ; not EOF
   adiw zl, 8                            ; goto buffer status
   ld r23, Z
 * sbr r23 , dmEOF
   st Z+, r23
   clr r17                               ; reset sector overflow
   rjmp _FileReadByte3
_FileReadByte1:
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileBuffer
   ;_FileReadByte2:
   eor r17, r25                          ; build sign for sector overflow
_FileReadByte3:
   andi r25, &H01                        ; clear range >= 512
   add zl, r24                           ; build pointer to byte in sector-buffer
   adc zh, r25
   ld r24, Z                             ; get byte
   lsr r17                               ; check Position-Bit; if <> 0 then sector overflow
   breq _FileReadByte9                   ; no sector overflow
   push r24                              ; save byte
```

```
   rcall _LoadNextFileSector
   pop r24
   ret
_FileReadByte9:
   rjmp _ClearDOSError
   ; Error handling
_FileReadByte81:
 * ldi r25 , cpEndOfFile
   ;_FileReadByte89:
   rjmp _SetDOSError
_FileReadByte98:
 * ldi r25 , cpFileAccessError
   rjmp _SetDOSError
[end]


[_CreateFile]
$EXTERNAL _CleanHWStack , _ClearDirEntry , _ClearDOSError , _DeleteFile , _GetFreeDirEntry , _Mem_Copy
$EXTERNAL _SearchDirEntry , _SetBitFileStatusYZ , _SetDirFATWritePending , _SetDirTimeDate
$EXTERNAL _SetDOSError , _SetFileHandle
   ;---------------------------------------------------------------------------
   ; Create File for
   ; o Output: start at 0, delete existing file
   ; o Append: start at file-End
   ; o Binary: start at 0
   ;
   ; Input: Z: Filename + extend
   ;        r24: File-number
   ;        r25: File-open-mode
   ;
   ; different entries:
   ; _CreateFile: exits with error if same file name exists on drive
   ; _CreateFile_CheckedNotExists: No checking for existing same file name in this routine
   ;                               is done, so it must be checked before
   ; _CreateFile_DeleteOld: Delete an existing file with same file name as new one
   ;---------------------------------------------------------------------------
_CreateFile_CheckedNotExists:
   push zh
   push zl
   push r24
   push r25
   rjmp _CreateFile2
_CreateFile_DeleteOld:
   push zh                              ; save pointer to File name
   push zl
   push r24
   push r25
   rcall _DeleteFile                    ; look for file and delete it, if exists
   brcc _CreateFile2                    ; no Error at File deleting
 * cpi r25 , cpFileNotFound             ; this Error is OK
   brne _CreateFile94                   ; other Error?
   rjmp _CreateFile2
_CreateFile:
   push zh                              ; Pointer to filename
   push zl
   push r24                             ; Save file-number to stack
   push r25                             ; save open mode to stack
   rcall _SearchDirEntry
   brcc _CreateFile91                   ; File-Name exists?
 * cpi r25 , cpFileNotFound
   brne _CreateFile94                   ; Only FileNotFound is OK
_CreateFile2:
   rcall _GetFreeDirEntry
   brcs _CreateFile92
   rcall _ClearDirEntry
   pop r25                              ; File Open mode
   pop r24                              ; File#
   push zh                              ; save Dir Entry
   push zl                              ;
   rcall _SetFileHandle
   pop xl
   pop xh
   brcs _CreateFile93                   ; no free file handle
```

```
   ; filename , attribut in DIR Entry
   st -y, zh                           ; Save File Handle Address
   st -Y, zl
   pop zl                              ; Pointer to filename
   pop zh
   push r25                            ; file open mode
   ldi r24, 11
   rcall _Mem_Copy                     ; copy filename to Dir Entry
   pop r25                             ; file open mode
   andi r25, &H01                      ; only read-only bit
   ori r25, &H20                       ; set archiv bit
   st X+, r25
   ; now X points to DIR-Entry+12
   adiw xl, 2                          ; goto create time/date
   rcall _SetDirTimeDate               ; writes 4 bytes
   adiw xl, 4                          ; goto Write time/date
   rcall _SetDirTimeDate
   rcall _SetDirWritePending
   rcall _SetBitFileEmptyYZ
   ld zl, Y+                           ; leave with file handle address in Z
   ld zh, Y+
   rjmp _ClearDOSError

_CreateFile91:
 * ldi r25 , cpFileExists
   rcall _SetDOSError
_CreateFile94:
   sec
_CreateFile92:
   rjmp _CleanHWStack4
_CreateFile93:
   rjmp _CleanHWStack2
[end]


[_ClearDirEntry]
   ;---------------------------------------------------------------------------
   ; clears a Dir Entry in the DIR buffer
   ; input settings: Z points to DIR Entry in Buffer
   ;---------------------------------------------------------------------------
_ClearDirEntry:
   ldi r24, &HE5
   st Z+, r24                          ; leave first char unchanged
   clr r24
   ldi r25, 31                         ; 32 BYtes to clear
_ClearDirEntry1:
   st Z+, r24
   dec r25
   brne _ClearDirEntry1
   sbiw zl, 32                         ; restore Dir Entry Pointer
   ret
[end]


[_DeleteFile]
$EXTERNAL _CheckA_Zero , _CheckFileInUse , _CheckUpdateFreeCluster , _CheckUpdateFreeDirEntry
$EXTERNAL _GetNextCluster , _SaveSector_DIR , _SaveSector_FAT , _SearchDirEntry
$EXTERNAL _SetDirFATWritePending , _SetDOSError , _SetFileNameError
   ;---------------------------------------------------------------------------
   ; Delete a file from the drive/disk
   ; no wildcard in name allowed
   ; Input: normalized file name -> Z
   ;---------------------------------------------------------------------------
_DeleteFile:
   ; entry point with previous checking of wildcard of filename with NormFileName
   breq _DeleteFile0
   rjmp _SetFileNameWildCardError0     ; Set DOSError for Wildcard in Filename
_DeleteFile0:
   ; in Z File-name + Extend           ; filename must be check for wildcard here
   rcall _SearchDirEntry
   brcs _DeleteFile4                   ; Not Found?
   ld r24, Z+11                        ; load file attribut
 * cbr r24, cpFileDeleteAllowed        ; deleting of this file type possible?
```

```
   breq _DeleteFile0b
 * ldi r25 , cpFileDeleteNotAllowed
   rjmp _SetDOSError
_DeleteFile0b:
   push zh                              ; save pointer to Dir-Entry
   push zl
   rcall _CheckFileInUse                ; check for file is use by program
   pop zl
   pop zh                               ; Error -> exit
   brcs _DeleteFile4

   push zh
   push zl
   rcall _CheckUpdateFreeDirEntry
   pop zl
   pop zh
_DeleteFile8:
   ldi r24, &HE5
   st Z, R24                            ; set first char of Dir Entry to deleted
   rcall _SetDirWritePending
   clr r18
   clr r19
   ld r16, Z+26                         ; Load First Cluster#
   ld r17, Z+27
   lds r0, {gbFileSystemStatus}         ; Info about FAT16/32
  * sbrs r0 , dFileSystemStatusFAT
   rjmp _DeleteFile0a
   ld r18, Z+20
   ld r19, Z+21
_DeleteFile0a:
   rcall _CheckA_Zero
   breq _DeleteFile3
_DeleteFile1:
   rcall _CheckUpdateFreeCluster        ; $$$$ prüfen ob nach getnextcluster besser
   rcall _GetNextCluster
   ldi r24, 0
   st Z+, r24
   st Z+, r24
   brtc _DeleteFile1b
   st Z+, r24
   st Z+, r24
_DeleteFile1b:
   rcall _SetFATWritePending            ; it do not affect C-Flag or r25
   brcc _DeleteFile1
 * cpi r25 , cpNoNextCluster            ; normal end of routine , go on
   brne _DeleteFile5                    ; other Error?
*#IF cSepFATHandle = 1
   rcall _SaveSector_FAT
   brcs _DeleteFile4
   rjmp _SaveSector_Dir
*#ELSE
   rjmp _SaveSector_FAT
*#ENDIF
_DeleteFile3:
   rjmp _SaveSector_Dir
_DeleteFile5:
   sec
_DeleteFile4:
   ret
[end]
```

```
[_CheckUpdateFreeDirEntry]
$EXTERNAL _CompareRange
   ;--------------------------------------------------------------------------------
   ; check, whether a DirEntry# freed within a file deleting routine is lower
   ; then just stored gwFreeDirEntry, and update gwFreeDirEntry if lower
   ;--------------------------------------------------------------------------------
_CheckUpdateFreeDirEntry:
   loadadr gwDirEntry , Z
   Loadadr gwFreeDirEntry , X
   ldi r24, 6                          ; DirEntry & Sector#
   rjmp _CompareRange_Update
[end]


[_CheckUpdateFreeCluster]
$EXTERNAL _CompareRange
   ;--------------------------------------------------------------------------------
   ; check, whether a Cluster# freed within a file deleting routine is lower
   ; then just stored glLastSearchedCluster, and update glLastSearchedCluster if lower
   ; Input: A Cluster# which gets free
   ;--------------------------------------------------------------------------------
_CheckUpdateFreeCluster:
   ldi zl, 16                          ; clusternumber to check is in A
   clr zh
   loadadr glLastSearchedCluster , X
   ldi r24, 4                          ; Cluster#
   rjmp _CompareRange_Update
[end]


[_CheckAvailableSectorSize]
$EXTERNAL _ClearBitFileStatusYZ , _ClearDOSError , _LoadFilePositionYZ2B , _LoadFileSizeYZ2A
$EXTERNAL _SetBitFileStatusYZ , _SetDOSError
$EXTERNAL _SUB32
   ;--------------------------------------------------------------------------------
   ; Check: Available Bytes in File-Sector
   ; Input:  Y0/1: FileHandle
   ; Output: R16,17 available Bytes
   ;         Bufferstatus: set EOF in Sector
   ;         r24: file/buffer status
   ;--------------------------------------------------------------------------------
_CheckAvailableSectorSize:
   rcall _LoadFileSizeYZ2A             ; load File Size to A
   rcall _LoadFilePositionYZ2B
  call _Sub32
   brmi _CheckAvailableSectorSize9
   breq _CheckAvailableSectorSize10    ; one pos. behind last byte
   ;_CheckAvailableSectorSize1:
   or r19, r18                         ; upper 2 MSB = 0 ?
   brne _CheckAvailableSectorSize6     ; to much inside
   ldi r20, 1                          ; 513,  &H201 to B
   ldi r21, 2
   cp r16, r20                         ; comp16
   cpc r17, r21
   brcs _CheckAvailableSectorSize8
_CheckAvailableSectorSize6:
   ldi r16, 0                          ; full sector: 512 Bytes available
   ldi r17, 2
 * ldi r24 , dmEOFAll
   rcall _ClearBitFileStatusYZ
   rjmp _ClearDOSError
_CheckAvailableSectorSize8:
 * ldi r24 , dmEOFinSector
   rcall _SetBitFileStatusYZ
   rjmp _ClearDOSError
_CheckAvailableSectorSize9:
 * ldi r25 , cpFilePositionError
   rjmp _CheckAvailableSectorSize11
_CheckAvailableSectorSize10:
 * ldi r25 , cpEndOfFile
_CheckAvailableSectorSize11:
 * ldi r24 , dmEOFALL
```

```
   rcall _SetBitFileStatusYZ
   rjmp _SetDOSError
[end]


[_CheckFileInUse]
$EXTERNAL _ClearDOSError , _CompareRange , _SetDOSError
   ;---------------------------------------------------------------------------
   ; check, whether a file is in use by BASCOM
   ; Input:  r20,21 DIR-Entry Number
   ; If File in use, then C-Flag set, Z points to filehandle
   ;---------------------------------------------------------------------------
_CheckFileInUse:
 * ldi r23 , cFileHandles                ; max. Number of file -handles
   loadadr abFileHandles(1) , Z
_CheckFileInUse1:
   ld r24, Z                             ; file#
   tst r24                               ; is a file entry there?
   breq _CheckFileinUse2
   push zh
   push zl
 * adiw zl, co_FileDirEntry
   loadadr gwDirEntry , X
   ldi r24, 6                            ; check DirEntry (2) and DirSector# (4)
   rcall _CompareRange
   pop zl
   pop zh
   breq _CheckFileInUse3
_CheckFileInUse2:
   dec r23
   breq _CheckFileInUse4                 ; All filehandle searched
 * subi zl , lByte(c_FileHandleSize_M)   ; advance filehandle pointer to next block
 * sbci zh , hbyte(c_FileHandleSize_M)
   rjmp _CheckFileInUse1
_CheckFileInUse3:
 * ldi r25 , cpFileInUse                 ; Not always an error , but easier this way
   rjmp _SetDOSError
_CheckFileInUse4:
   rjmp _ClearDOSError
[end]


[_AllocateNextFileSector]
$EXTERNAL _ClearDOSError , _ClearFileSector , _Cluster2Sector , _ClusterA2FAT , _GetFreeCluster
$EXTERNAL _GetNextCluster , _GetNextSectorNumber , _GetSector , _LoadFileBufferStatusYZ
$EXTERNAL _LoadFileSectorNumberYZ2A , _SaveFileSectorNumberA2YZ , _SetDirFATWritePending
$EXTERNAL _SetDOSError
   ;---------------------------------------------------------------------------
   ; Allocate a new sector to a file
   ; Input:  Y0-> Filehandle
   ; OutPut: Sector# in A and FileHandle->Sector#
   ;---------------------------------------------------------------------------
_AllocateNextFileSector:
   rcall _LoadFileSectorNumberYZ2A       ; current sector# + 1
   rcall _GetNextSectorNumber
   brcs _AllocateNextFileSector4         ; No next cluster available ?
_AllocateNextFileSector3:
  rcall _LoadFileBufferStatusYZ          ; buffer status to _temp1 (r24)
 * sbrs r24 , dEOF                       ; only clear Sectorbuffer if EOF
   rjmp _GetSector_File                  ; load sector
   rcall _SaveFileSectorNumberA2YZ
   rjmp _ClearFileSector                 ; clears also C-Flag for no Error

_AllocateNextFileSector4:
 * cpi r25 , cpNoNextCluster             ; this error is ok
   brne _AllocateNextFileSector92        ; other Error?
   rcall _GetFreeCluster
   brcs  _AllocateNextFileSector92
   ldi r24, &HFF
   st Z+, r24                            ; set to last Cluster#
   st Z+, r24
```

```
   lds r0, {gbFileSystemStatus}
 * sbrs r0 , dFileSystemStatusFAT
   rjmp _AllocateNextFileSector5
   st Z+, r24
   ldi r24, &H0F
   st Z+, r24
_AllocateNextFileSector5:
   rcall _SetFATWritePending
   push r16                              ; save new Cluster#
   push r17
   push r18
   push r19
   rcall _GetNextCluster_Y               ; Set Z to old Cluster Entry
   ; Check C-Flag, if Set and cpNoNextCluster, than OK
   pop r19
   pop r18
   pop r17
   pop r16
   brcc _AllocateNextFileSector91        ; Error cpNoNextCluster must be set
 * cpi r25 , cpNoNextCluster
   brne _AllocateNextFileSector91
   rcall _ClearDOSError
   rcall _ClusterA2FAT
   rcall _Cluster2Sector
   rjmp _AllocateNextFileSector3

_AllocateNextFileSector91:
   ; Error in Cluster Handling
 * ldi r25 , cpClusterError
   rjmp _SetDOSError
_AllocateNextFileSector92:
   ret
[end]


[_AllocateFirstFileSector]
$EXTERNAL _ClearA , _ClearBitFileStatusYZ , _Cluster2Sector , _GetDirEntry , _GetFreeCluster
$EXTERNAL _SaveFileFirstClusterA2YZ , _SaveFilePositionA2YZ , _SaveFileSectorNumberA2YZ
$EXTERNAL _SetDirFATWritePending
   ;---------------------------------------------------------------------------
   ; Allocate first sector of file
   ; entry with filehandle in Y
   ;---------------------------------------------------------------------------
_AllocateFirstFileSector:
   rcall _GetFreeCluster
   brcs  _AllocateFirstFileSector91
   ldi r24, &HFF                         ; Write "Last Cluster"
   st Z+, r24
   st Z+, r24
   lds r0, {gbFileSystemStatus}
 * sbrs r0 , dFileSystemStatusFAT
   rjmp _AllocateFirstFileSector1
   st Z+, r24
   ldi r24, &H0F
   st Z+, r24
_AllocateFirstFileSector1:
   rcall _SetFATWritePending
   push r19                              ; Save Cluster#
   push r18
   push r17
   push r16
   rcall _SaveFileFirstClusterA2YZ       ; leave file handle base in Z
   rcall _GetDirEntry_Y
   pop r16                               ; first Cluster#
   pop r17
   pop r18
   pop r19
   brcs _AllocateFirstFileSector91
   std Z+26, r16                         ; write first cluster# to DirEntry
   std Z+27, r17
   lds r0, {gbFileSystemStatus}
 * sbrs r0 , dFileSystemStatusFAT
   rjmp _AllocateFirstFileSector2
```

```
   std Z+20, r18
   std Z+21, r19
_AllocateFirstFileSector2:
   rcall _SetDirWritePending
   rcall _Cluster2Sector
   rcall _SaveFileSectorNumberA2YZ
   rcall _ClearA
   rcall _SaveFilePositionA2YZ
 * ldi r24 , dmFileEmpty                  ; Clear File Empty
   rcall _ClearBitFileStatusYZ
   clc
   ; End of Routine
_AllocateFirstFileSector91:
   ret
[end]


[_SaveFileSector]
$EXTERNAL _A2LongX , _AddrFileSectorNumber2Z , _ClearBitFileStatusYZ , _ClearDOSError
$EXTERNAL _DriveWriteSector_SEV , _GetDirEntry , _LoadFileBufferStatusYZ , _LoadFileSizeYZ2A
$EXTERNAL _SaveSector_DIR , _SaveSector_FAT , _SetDirFATWritePending , _SetDirTimeDate
   ;---------------------------------------------------------------------------
   ; save a file-sector and update the directory-entry and FAT if nesseccary
   ; input: file handle address in Y+0/1
   ;---------------------------------------------------------------------------
_SaveFileSector:
   rcall _LoadFileBufferStatusYZ          ; buffer status to _temp1
 * sbrs r24 , dWritePending
   rjmp _SaveFileSector2                  ; Sector is to write
   ;_SaveFileSector1:
   ; part for saving sector
   ; prepare for writing
   rcall _AddrFileSectorNumber2Z          ; address of sector# to Z
   mov xl, zl
   mov xh, zh
   adiw zl, 4                             ; order of sectornumber / Bufferstatus / Buffer is fixed
   ld r24, Z                              ; load buffer status
 * cbr r24 , dmWritePending               ; clear write pending
   st Z+, r24                             ; save new bufferstatus and set Z to buffer
   rcall _DriveWriteSector_SEV
   brcs _SaveFileSector9
   ; check for file size
_SaveFileSector2:
   rcall _LoadFileBufferStatusYZ          ; buffer status to r24
 * sbrs r24 , dFATDirWritePending
   rjmp _SaveFileSector10
 * sbrs r24 , dFATDirSaveAtEnd            ; not save any time at file sector savePrint
   rjmp _SaveFileSector3
 * sbrs r24 , dFATDirSaveAnyWay
   rjmp _SaveFileSector10
_SaveFileSector3:
   rcall _GetDirEntry_Y
   brcs _SaveFileSector9
   mov xl, zl
   mov xh, zh
   adiw xl, 11                            ; set X-pointer to Directory Status Byte
   ld r24, X
   sbr r24, &B00100000                    ; set archiv bit
   st X, r24
   adiw xl, 17                            ; next if file size
 * ldi r24 ,(dmFATDirWritePending + dmFATDirSaveAnyWay)
   rcall _ClearBitFileStatusYZ
   rcall _LoadFileSizeYZ2A
   rcall _A2LongX
   sbiw xl, 10                            ; go to Dir-Entry Time/Date
   rcall _SetDirTimeDate
   rcall _SetDirWritePending
*#IF cSepFATHandle = 1
   rcall _SaveSector_Dir
   brcs _SaveFileSector9
   rjmp _SaveSector_FAT
*#ELSE
   rjmp _SaveSector_Dir
```

```
*#ENDIF
_SaveFileSector9:
   ret
_SaveFileSector10:
   rjmp _ClearDOSError
[end]


[_ClearFileSector]
$EXTERNAL _Add32_Word , _AddrFileBuffer2Z , _ClearDOSError , _Cluster2Sector , _GetSector , _LongZ2A
$EXTERNAL _SetDirFATWritePending , _Sub32_1
   ;---------------------------------------------------------------------------
   ; Clear file sector buffer
   ; Input settings: File handle address in Y+0/1
   ;---------------------------------------------------------------------------
_ClearDirCluster:
   ; Entry with Cluster# in A
   rcall _Cluster2Sector
   lds r24, {gbSectorsPerCluster}
_ClearDirCluster1:
   push r24
   rcall _GetSector_Dir
   brcs _ClearDirCluster2
   rcall _ClearSector
   rcall _SetDirWritePending
   clc
_ClearDirCluster2:
   pop r24
   brcs _ClearDirCluster9
   dec r24
   breq _ClearDirCluster7
   loadadr glDirSectorNumber , Z
   rcall _LongZ2A
   rcall _Add32_1
   rjmp _ClearDirCluster1
_ClearDirCluster7:
   ; go back to first sector of cluster
   lds r20, {gbSectorsPerCluster}
   dec r20
   breq _ClearDirCluster8
   loadadr glDirSectorNumber , Z
   rcall _LongZ2A
   sub r16 , r20
   rcall _Sub32_Byte
   rjmp _GetSector_Dir
_ClearDirCluster8:
   rjmp _ClearDOSError
_ClearDirCluster9:
   ret

_ClearFileSector:
   rcall _AddrFileBuffer2Z              ; address of File Buffer to Z
_ClearSector:
   clr r23
   ldi r25, 2                           ; 512 byte counter
   ldi r24, 0
_ClearFileSector1:
   st Z+, r23
   sbiw r24, 1
   brne _ClearFileSector1
   subi zh, 2                           ; set back to buffer-start
   rjmp _ClearDOSError
[end]
```

```
[_SaveSector_FAT]
$EXTERNAL _DriveWriteSector_SEV , _SaveSector2FATCopy
   ;---------------------------------------------------------------------------
   ; save FAT sector in FAT sector buffer to Drive/disk if necessary
   ; make FAT copy too, if 2 FAT Copies in File system
   ;---------------------------------------------------------------------------
_SaveSector_FAT:
 * ldi _temp1 , cp_FATBufferInitStatus
*#IF cSepFATHandle = 1
   loadadr glFATSectorNumber , X         ; Sector# to X
   loadadr gbFATBufferStatus , Z
*#ELSE
   loadadr glDIRSectorNumber , X         ; Sector# to X
   loadadr gbDIRBufferStatus , Z
*#ENDIF
   ld _temp2, Z                          ; load buffer status and brings Z to buffer start
 * sbrs _temp2 , dWritePending           ; Must sector written be back
   ret
   ; Z is now at buffer start
   st Z+, _temp1                         ; save new bufferstatus
   push xh
   push xl
   rcall _DriveWriteSector_SEV
   pop xl
   pop xh
   brcc _SaveSector_FAT1
   ret                                   ; return on error
_SaveSector_FAT1:
   rjmp _SaveSector2FATCopy
[end]


[_SaveSector_DIR]
$EXTERNAL _DriveWriteSector_SEV
   ;---------------------------------------------------------------------------
   ; Save DIR sector in DIR sector buffer to Drive/Disk if necessary
   ;---------------------------------------------------------------------------
_SaveSector_DIR:
 * ldi _temp1 , cp_DIRBufferInitStatus
   loadadr glDIRSectorNumber , X         ; Sector# to X
   loadadr gbDIRBufferStatus , Z
   ld _temp2, Z                          ; load buffer status and brings Z to buffer start
 * sbrs _temp2 , dWritePending           ; Must sector written be back
   ret
   ; Z is now at buffer start
   st Z+, _temp1                         ; save new bufferstatus
   rjmp _DriveWriteSector_SEV
[end]


[_SaveSector2FATCopy]
$EXTERNAL _A2LongX , _DriveWriteSector_SEV , _LongX2A , _LongZ2B
$EXTERNAL _ADD32
   ;---------------------------------------------------------------------------
   ; make FAT Copy, if file system have two FATs
   ;---------------------------------------------------------------------------
   ; X-> Sector# of actual FAT sector
_SaveSector2FATCopy:
*#IF cFATSecondUpdate = 0
   ret                                   ; make no FAT-COPY if configured
*#ENDIF
   loadadr gbNumberOfFATs , Z
   ld _temp1, Z+                         ; Number of FATs
   sbrs _temp1, 1                        ; 2 FATs ? (Only 1 or 2 supported)
   ret
   rcall _LongZ2B                        ; Sector per FAT, behind gbNumberOfFATs in SRAM
   rcall _LongX2A                        ; load sector in first FAT
   mov zl, xl                            ; now buffer -1
   mov zh, xh
   adiw zl , 1                           ; now to buffer
   call _Add32                           ; first FAT Sector + Sectors per FAT
   loadadr glFS_Temp1 , X                ; temporary hold sector# of FAT -copy
```

```
   rcall _A2LongX                         ; Sector number of second FAT
   sbiw xl, 4                             ; X-> to sector#
   rjmp _DriveWriteSector_SEV
[end]


[_CheckFileAccess_Mode]
$EXTERNAL _SetDOSError
   ;---------------------------------------------------------------------------
   ; check, whether wanted operation suits to File-Open Mode
   ; Constant for Operation is passed in r25
   ;---------------------------------------------------------------------------
_CheckFileAccess_Mode:
   ldd zl, Y+0
   ldd zh, Y+1
_CheckFileAccess_Mode_Z:
   mov r0, r25                            ; save to r0, scratch register
 * ldd r25 , Z + co_FileMode
   and r25, r0                            ; if one bit match, than access allowed
   breq _CheckFileAccess_Mode2
   clc
   ret
_CheckFileAccess_Mode2:
 * ldi r25 , cpFileAccessError
   rjmp _SetDOSError
[end]


[_FileWriteByte]
$EXTERNAL _A2LongZ , _Add32_Word , _AllocateFirstFileSector , _AllocateNextFileSector
$EXTERNAL _ClearFileSector , _LongZ2A , _LongZ2B , _SaveFileSector , _SetDOSError
   ;---------------------------------------------------------------------------
   ; write one byte to file and do all checks for empty file, passing to new
   ; sector, cluster
   ; Input:  Y0-> File handle
   ;         r24: Byte to write
   ;---------------------------------------------------------------------------
_FileWriteByte:
   ; check access mode
   ldd zl, Y+0
   ldd zh, Y+1
 * ldd r25 , Z + co_FileMode
 * andi r25 , cFileWrite_Mode
   breq _FileWriteByte98                  ; not allowed File mode
_FileWriteByte0:
   ldd zl, Y+0
   ldd zh, Y+1
   push r24                               ; save byte to write to stack
 * ldd r25 , Z + co_FileBufferStatus      ; check buffer status
 * sbrs r25 , dFileEmpty
   rjmp _FileWriteByte2
   ; File is empty
   rcall _AllocateFirstFileSector
   brcs _FileWriteByte99                  ; Error ?
   rcall _ClearFileSector
   ldd zl, Y+0
   ldd zh, Y+1
 * ldd r25 , Z + co_FileBufferStatus      ; file bufferstatus
_FileWriteByte2:
 * adiw zl , co_FilePosition
   rcall _LongZ2A                         ; load File Position
   mov r15, r17                           ; save sector-part in position (bit0-8 of position)
   mov r14, r16
   rcall _Add32_1                         ; Advance file position + 1
   sbiw zl, 4                             ; back to file position
   rcall _A2LongZ                         ; save new file position
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileSize
   rcall _LongZ2B                         ; load file size to B
   cp r20, r16                            ; compare size - position
   cpc r21, r17                           ; if size < position than adjust size to pos
```

```
   cpc r22, r18
   cpc r23, r19
   brcc _FileWriteByte3
   sbiw zl, 4                           ; set pointer to File Size
   rcall _A2LongZ                       ; save File position to File size
 * sbr r25 ,(dmEOF + dmFATDirWritePending)   ; set End of File and write of FAT / Dir for this file is
                                          pending
_FileWriteByte3:
   eor r17, r15                         ; build check for Sector overflow
 * sbr r25 , dmWritePending             ; Set for saving sector
   ld zl, Y+0
   ld zh, Y+1
 * adiw zl , co_FileBufferStatus
   st Z+, r25                           ; save new buffer status and advance to buffer start
   mov r25, r15                         ; r15 not allow andi
   andi r25, &H01                       ; mask Bits over 512
   add zl, r14
   adc zh, r25
   ;_FileWriteByte4:
   pop r24                              ; Byte to write from stack
   st Z+, r24                           ; write byte to buffer
   lsr r17                              ; 0, if no overflow
   breq _FileWriteByte5
   rcall _SaveFileSector                ; go with old file position to saveSector
   rjmp _AllocateNextFileSector

_FileWriteByte98:
 * ldi r25 , cpFileAccessError
   rjmp _SetDOSError

_FileWriteByte99:
   pop r24
   ret
   ; clean stack
_FileWriteByte5:
   clc
   ret
[end]


[_GotoFilePosition]
$EXTERNAL _Add32_Word , _AllocateNextFileSector , _CheckEOF , _ClearDOSError , _ClearFileSector
$EXTERNAL _Cluster2Sector , _GetNextCluster , _GetSector , _LoadFileBufferStatusYZ
$EXTERNAL _LoadFileFirstClusterYZ2A , _LoadFilePositionYZ2A , _LoadFileSectorNumberYZ2A
$EXTERNAL _LoadFileSizeYZ2B , _Position2SeqCluster , _SaveFilePositionA2YZ , _Sector2Cluster
$EXTERNAL _SetDOSError , _Sub32_1
$EXTERNAL _COMP32
   ;---------------------------------------------------------------------------
   ; position the read/write pointer of a file to a new position
   ; Input:  new Position in r16-r19
   ;         new position is at:
   ;                 _GotoFilePosition: 0-based
   ;                 _GotoFilePosition_1Base: 1-based
   ;---------------------------------------------------------------------------
_GotoFilePosition_1Base:
   rcall _Sub32_1
_GotoFilePosition:
   ;check for invalid file position Long-value = minus
   tst r19                              ; MSB bit set if LONG value is minus
   brpl _GotoFilePosition02
 * ldi r25 , cpInvalidFilePosition
   rjmp _SetDOSError
_GotoFilePosition02:
   brne _GotoFilePosition01             ; new position pointer > 0 ?
   rcall _LoadFileBufferStatusYZ
 * sbrc r24 , dFileEmpty
   rjmp _ClearDOSError                  ; if empty do nothing; so sector to load
   ; check of new position > size
_GotoFilePosition01:
   rcall _LoadFileSizeYZ2B              ; load file size to A
  call _Comp32
   breq _GotoFilePositionEOF            ; check for skip over sector border
   brcc _GotoFilePosition9              ; Error? position to high
_GotoFilePosition1:
```

```
    push _A4                            ; new position temp. to stack
    push _A3
    push _A2
    push _A1
    rcall _Position2SeqCluster          ; calculate with new file position
    mov r21, r12                        ; cluster sequence number
    mov r22, r13
    mov r23, r14
    push r20                            ; sector offset in cluster
    rcall _LoadFilePositionYZ2A         ; load file position to A
    rcall _Position2SeqCluster          ; caluculate with old file position
    rcall _LoadFileSectorNumberYZ2A
    push r23
    push r22
    push r21
    rcall _Sector2Cluster
    pop r21
    pop r22
    pop r23
    cp r21, r12                         ; new - old
    cpc r22, r13
    cpc r23, r14
    brcs _GotoFilePosition2             ; lower cluster?
    breq _GotoFilePosition4
    ; same cluster?
    ; new cluster higher than old on
    sub r21 , r12                       ; build difference
    sbc r22, r13
    sbc r23, r14
    rjmp _GotoFilePosition3
_GotoFilePosition2:
    rcall _LoadFileFirstClusterYZ2A
_GotoFilePosition3:
    subi r21, 1
    sbci r22, 0
    sbci r23, 0
    brcs _GotoFilePosition4
    push r23                            ; save count of Cluster to skip
    push r22
    push r21
    rcall _GetNextCluster
    pop r21
    pop r22
    pop r23
    brcc _GotoFilePosition3
_GotoFilePosition4:
    rcall _Cluster2Sector
    pop r20                             ; sector in Cluster
    rcall _Add32_Byte                   ; add sector-offset in cluster now in A
    rcall _GetSector_File               ; load new Sector
    brcs _GotoFilePositionEOF9          ; Error?
    pop _A1                             ; new file position back from stack
    pop _A2
    pop _A3
    pop _A4
    rcall _SaveFilePositionA2YZ         ; save new file position
    rcall _CheckEOF                     ; set EOF bit to new position situation
    rjmp _ClearDOSError
_GotoFilePosition9:
  * ldi r25 , cpFilePositionError
    rjmp _SetDOSError

_GotoFilePositionEOF:
    ; for skip over sector border
    tst r16                             ; test for skip over sector border
    brne _GotoFilePosition1             ; bit 0-8 of position = 0
    mov r24, r17
    andi r24, &H01                      ; test bit 8 of position
    brne _GotoFilePosition1
    ; Reduce pointer to last byte
    rcall _Sub32_1
    rcall _GotoFilePosition             ; do it with position of last byte
    brcs _GotoFilePositionEOF9          ; Error? Exit
    rcall _LoadFilePositionYZ2A         ; load file position to A
```

```
   rcall _Add32_1
   rcall _SaveFilePositionA2YZ
   rcall _AllocateNextFileSector
   brcc _GotoFilePositionEOF1
 * cpi r25 , cpEndOfFile                 ; EOF is OK
   breq _GotoFilePositionEOF1
   sec
_GotoFilePositionEOF9:
   ret
_GotoFilePositionEOF1:
   rcall _LoadFileBufferStatusYZ
 * sbrc r24 , dEOF                       ; only clear Sectorbuffer if EOF
   rjmp _ClearFileSector                 ; clears also C-Flag for no Error
   rjmp _ClearDOSError


[end]



[_FileWriteString]
$EXTERNAL _FileWriteByteArray
   ;---------------------------------------------------------------------------
   ; write a string to a file
   ; Input: Y0/1 -> Filehandle
   ;         X-> String to write
   ;---------------------------------------------------------------------------
_FileWriteString:
   push xh                               ; save Start of String
   push xl
   clr r20                               ; clear Byte-counter
   clr r21
_FileWriteStringLength1:
   ld r24, X+                            ; get byte from string
   tst r24                               ; string -terminator?
   breq _FileWriteStringLength2          ; ready string-length testing
   subi r20, &HFF                        ; + 1
   sbci r21, &HFF
   brne _FileWriteStringLength1
_FileWriteStringLength2:
   pop xl                                ; address of string-start back to X
   pop xh
   rjmp _FileWriteByteArray
[end]



[_FileWriteStringQuotationMark]
$EXTERNAL _FileWriteQuotationMark , _FileWriteString
   ;---------------------------------------------------------------------------
   ; write a string surrounded with quotation marks to a file
   ; Input: Y0/1 -> Filehandle
   ;         X-> String to write
   ;---------------------------------------------------------------------------
   ; Y0/1 File handle
   ; X-> String
_FileWriteStringQuotationMark:
*#IF cTextQuotationMarks = 1
   push xh
   push xl
   rcall _FileWriteQuotationMark
   pop xl
   pop xh
   brcs _FileWriteStringQuotationMark1
*#ENDIF
   rcall _FileWriteString
*#IF cTextQuotationMarks = 1
   brcs _FileWriteStringQuotationMark1
   rjmp _FileWriteQuotationMark
*#ENDIF
_FileWriteStringQuotationMark1:
   ret
[End]
```

```
[_FileWriteCRLF]
$EXTERNAL _FileWriteByte
   ;---------------------------------------------------------------------------
   ; write a CR+LF to a file (0D0A)
   ; Y0/1 -> File Handle
   ;---------------------------------------------------------------------------
_FileWriteCRLF:
   ldi r24, 13
   rcall _FileWriteByte
   brcs _FileWriteCRLF1
   ldi r24, 10
   rjmp _FileWriteByte
_FileWriteCRLF1:
   ret
[end]


[_FileWriteComma]
$EXTERNAL _FileWriteByte
   ;---------------------------------------------------------------------------
   ; write a Comma (field separator) to a file (,)
   ; Y0/1 -> File Handle
   ;---------------------------------------------------------------------------
_FileWriteComma:
 * ldi r24, cVariableSeparator           ; default is comma
   rjmp _FileWriteByte
[end]


[_FileWriteQuotationMark]
$EXTERNAL _FileWriteByte
   ;---------------------------------------------------------------------------
   ; write a QuotationMark for String Writing to a file
   ; Y0/1 -> File Handle
   ;---------------------------------------------------------------------------
_FileWriteQuotationMark:
   ldi r24, 34                           ; QuotationMark = &H22 = 34
   rjmp _FileWriteByte
[end]


[_FileWriteStrConst]
$EXTERNAL _FileWriteByte
$EXTERNAL _LPMBYTE
   ;---------------------------------------------------------------------------
   ; Write a Constant from FLASH to the file
   ; Y0/1 -> filehandle
   ; Z -> FLASH
   ;---------------------------------------------------------------------------
_FileWriteStrConst:
  call _LPMbyte                          ; get byte from code space
   Breq _FileWriteStrConstExit1
   Mov r24,r0
   push zh
   push zl                               ; exit
   rcall _FileWriteByte
   pop zl
   pop zh
   brcs _FileWriteStrConstExit2
   Rjmp _FileWriteStrConst
_FileWriteStrConstExit1:
   clc
_FileWriteStrConstExit2:
   Ret
[end]
```

```
[_FileWriteDecByte]
$EXTERNAL _FileWrite_StrBuf
$EXTERNAL _NUM2STR
   ;---------------------------------------------------------------------------
   ; Write a Byte variable in ASCII to a file
   ; Input: Z-> Byte variable
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWriteDecByte:
   loadadr ___strbuf , X
  call _PrintDecB
   rjmp _FileWrite_StrBuf
[end]


[_FileWriteDecWord]
$EXTERNAL _FileWrite_StrBuf
$EXTERNAL _NUM2STR
   ;---------------------------------------------------------------------------
   ; Write a Word variable in ASCII to a file
   ; Input: Z-> Word variable
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWriteDecWord:
   loadadr ___strbuf , X
  call _PrintWord
   rjmp _FileWrite_StrBuf
[end]


[_FileWriteDecInt]
$EXTERNAL _FileWrite_StrBuf
$EXTERNAL _NUM2STR
   ;---------------------------------------------------------------------------
   ; Write a Integer variable in ASCII to a file
   ; Input: Z-> Integer variable
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWriteDecInt:
   loadadr ___strbuf , X
  call _PrintsDecW
   rjmp _FileWrite_StrBuf
[end]


[_FileWriteDecLong]
$EXTERNAL _FileWrite_StrBuf
$EXTERNAL _NUM2STR
   ;---------------------------------------------------------------------------
   ; Write a Long variable in ASCII to a file
   ; Input: Z-> Long variable
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWriteDecLong:
   loadadr ___strbuf , X
  call _PrintLong
   rjmp _FileWrite_StrBuf
[end]


[_FileWriteDecSingle]
$EXTERNAL _FileWrite_StrBuf
$EXTERNAL _FSRAMA1
   ;---------------------------------------------------------------------------
   ; Write a Single variable in ASCII to a file
   ; Input: Z-> Single variable
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWriteDecSingle:
   loadadr ___strbuf , X
```

```
  call _FSRAMA1
  call _ftoa
   rjmp _FileWrite_StrBuf
[end]


[_FileWriteDecDouble]
$EXTERNAL _FileWrite_StrBuf
   ;---------------------------------------------------------------------------
   ; Write a Single variable in ASCII to a file
   ; Input: Z-> Double variable
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWriteDecDouble:
   loadadr ___strbuf , X
   clr r16                          ; clear appending Zeros
   call _Dbl_Bin2ASC_XZ
   rjmp _FileWrite_StrBuf
[end]


[_FileWrite_StrBuf]
$EXTERNAL _FileWriteString
   ;---------------------------------------------------------------------------
   ; write the string in ___strBuf to the file
   ;        Y0/1 -> file handle
   ;---------------------------------------------------------------------------
_FileWrite_StrBuf:
   loadadr ___strbuf , X
   rjmp _FileWriteString
[End]


[_FileReadString]
$EXTERNAL _FileReadByte
   ;---------------------------------------------------------------------------
   ; Read a String from a File
   ; File handle in Y0/1
   ; String pointer in X
   ; max String length in r20
   ; Check for string surrounding of quotation mark (QM) is done
   ; if string is surrounded of QM, String starts behind first QM and ends before next QM
   ; r25 is status-byte: bit0=1: reading first byte (check for QM)
   ;                     bit1=1: String is embedded with QM check every byte for end of string =
                                 second QM
   ;                     bit2=1: second QM reached (string ready) skipping behind next comma(,) or
                                 CRLF
   ;---------------------------------------------------------------------------
_FileReadString2StrBuf:
   loadadr ___strbuf , X
_FileReadString:
   ldi r25, 1                       ; Bit0=1: Flag for reading first byte
_FileReadString1:
   push r25
   push r20
   push xh
   push xl
   rcall _FileReadByte              ; read one byte from file
   pop xl
   pop xh
   pop r20
   pop r25
   brcs _FileReadString8            ; Error ?
   tst r25                          ; if all bit=0 then no check for QM
   breq _FileReadString5
   sbrs r25, 0                      ; checking first Byte?
   rjmp _FileReadString2
   cpi r24, 34                      ; is first Byte a QM
   brne _FileReadString4
   ldi r25, 2                       ; Bit1=1: Flag for String embedded in QM
   rjmp _FileReadString1            ; read next byte
```

```
_FileReadString2:
   sbrc r25, 2
   rjmp _FileReadString5                ; second Quotation mark just passed, check for Comma of CR /
                                         LF
   cpi r24, 34                          ; Quotation mark?
   brne _FileReadString6
   ldi r25, 4                           ; Bit2=1: Flag for second Quotation mark reached
   rjmp _FileReadString1                ; read next byte (Should be comma or CR / LF)
_FileReadString4:
   clr r25                              ; String is not surrounded from quotation marks
_FileReadString5:
 * cpi r24, ASC(",")                   ; default is comma(,) ?
   breq _FileReadString7                ;
 * cpi r24, cVariableSeparator          ; additional separator
   breq _FileReadString7                ;
   cpi r24, &H0D                        ; CR?
   breq _FileReadString1                ; ignore &H0D and get next char (should be &H0A)
   cpi r24, &H0A                        ; &H0A is End of Line reached
   breq _FileReadString7
_FileReadString6:
   tst r20                              ; Just at end of String, do not store any more
   breq _FileReadString1
   bst r25, 2                           ; do not store if second Quotation mark reached
   brts _FileReadString1
   st X+, r24                           ; store byte from file to string
   dec r20                              ; decr. String-length counter
   rjmp _FileReadString1                ; read next byte
_FileReadString7:
   clc                                  ; no error sign
_FileReadString8:
   ldi r24, 0                           ; string terminator
   st x, r24
   ret
[End]
```

```
[_FileReadDec2Num]
$EXTERNAL _FileReadString
$EXTERNAL _STR2NUM
   ;--------------------------------------------------------------------------------
   ; Read a Numeric Value in ASCII from File, convert it to Binary
   ; Input: File handle Y0/1
   ; Output: r16-r19
   ;--------------------------------------------------------------------------------
_FileReadDec2Num:
   Ldi r20,15                           ; use ___strBuf
   rcall _FileReadString2StrBuf
   loadadr ___strbuf , X
   jmp _Str2Num                         ; convert string to num
[end]
```

```
[_FileReadDecSingle]
$EXTERNAL _FileReadString
$EXTERNAL _ASC2FLOAT
   ;--------------------------------------------------------------------------------
   ; Read a Single in ASCII from File, convert it to Binary
   ; Input: File handle Y0/1
   ; Output: r13,r14,r15,r16 (FP-ACC1)
   ;--------------------------------------------------------------------------------
_FileReadDecSingle:
   Ldi r20,15
   rcall _FileReadString2StrBuf
   loadadr ___strbuf , X
   jmp _Asc2Float                       ; convert string to num
[end]
```

```
[_FileReadDecDouble]
$EXTERNAL _FileReadString
   ;--------------------------------------------------------------------------------
   ; Read a Double in ASCII from File, convert it to Binary
   ; Input: File handle Y0/1
   ;         Double-Variable pointer in Z for output
   ;--------------------------------------------------------------------------------
_FileReadDecDouble:
   push zh
   push zl
   Ldi r20,23
   rcall _FileReadString2StrBuf
   loadadr ___strbuf , X
   pop zl
   pop zh
   jmp _Dbl_ASC2Bin_zx                  ; convert string to Double
[end]



[_FileReadByteArray]
$EXTERNAL _AddrFileBuffer2Z , _CheckEOF , _CheckFileAccess_Mode , _ClearDOSError , _FileReadByte
$EXTERNAL _FileWriteByteArray , _LoadFileBufferStatusYZ , _LoadFilePositionYZ2A
$EXTERNAL _SaveFilePositionA2YZ , _SetBitFileStatusYZ , _SetDOSError
$EXTERNAL _COMP32
   ;--------------------------------------------------------------------------------
   ; Read a desired amount of bytes from file and store it to SRAM Byte
   ; Input: Y+0,1 Filehandle
   ;        X Pointer to SRAM where to write the bytes from file
   ;        r20,21 : number of bytes to transfer
   ;--------------------------------------------------------------------------------
_FileReadByteArray:
 * ldi r25 , cFileRead_Mode
   rcall _CheckFileAccess_Mode
   brcs _FileReadByteArray90
_FileReadByteArray0:
   rcall _LoadFileBufferStatusYZ        ; bring filehandle to Z too
 * sbrc r24 , dEOF
   rjmp _FileReadByteArray91            ; EOF
   push r21
   push r20
   rcall _CheckEOF                      ; leaves available bytes to EOF in A
   pop r20
   pop r21
   brcs _FileReadByteArray91            ; behind EOF
   clr r22
   clr r23
  call _Comp32
   brcc _FileReadByteArray2
   mov r20, r16                         ; remaining byte to counter
   mov r21, r17
_FileReadByteArray2:
   rcall _FilePosition2A_BufferByte
_FileReadByteArray3:
   subi r20, 1                          ; decr. byte counter
   sbci r21, 0
   brcs _FileReadByteArray80            ; last byte written
   cpi r16, &HFF                        ; at skip to next sector
   brne _FileReadByteArray5
   sbrs r17, 0                          ; if bit8 = 1 than realy sector skip
   rjmp _FileReadByteArray5             ; no sector skip
   ;_FileReadByteArray4:
   rcall _SaveFilePositionA2YZ          ; save file position from A
   push xh
   push xl
   push r21
   push r20
   rcall _FileReadByte0                 ; do all handle with next Sector
   pop r20
   pop r21
   pop xl
   pop xh
```

```
   brcs _FileReadByteArray90             ; Error ?
   rcall _LoadFilePositionYZ2A
   rcall _AddrFileBuffer2Z
   rjmp _FileReadByteArray7
_FileReadByteArray5:
   subi r16, &HFF                        ; +1
   sbci r17, &HFF
   ;_FileReadByteArray6:
   ld r24, Z+
_FileReadByteArray7:
   st X+, r24
   rjmp _FileReadByteArray3
_FileReadByteArray80:
   rcall _SaveFilePositionA2YZ           ; save file position from A
   rcall _CheckEOF                       ; checks an set EOF
   rjmp _ClearDOSError

_FileReadByteArray90:
   ret
_FileReadByteArray91:
   rcall _SetBitEOFYZ
 * ldi r25 , cpEndOfFile                 ; EOF
   rjmp _SetDOSError
[end]


[_Position2SeqCluster]
   ;---------------------------------------------------------------------------
   ; calculate from a file position the ordinal cluster# (Cluster# within file)
   ; and the ordinal number of the sector in the cluster
   ; routine is used from _GotoFilePosition
   ; Input:  A         - Position
   ; OutPut: r10,11,12 - Cluster sequence#
   ;         r20       - Sector in cluster
   ;---------------------------------------------------------------------------
_Position2SeqCluster:
   lds r24, {gbSectorsPerCluster}
   lsr r19
   ror r18
   ror r17                               ; shift out Bit 8 of position (256-511)
   mov r20, r17                          ; Low part of sector-sequence#
   ldi r25, 0                            ; mask for Sector in Cluster
_Position2SeqCluster2:
   lsr r24
   brcs _Position2SeqCluster3
   lsr r19
   ror r18
   ror r17
   sec                                   ; Set C-Flag to shift in Bit=1 in next rol
   rol r25
   rjmp _Position2SeqCluster2
_Position2SeqCluster3:
   mov r12, r17
   mov r13, r18
   mov r14, r19
   And r20, r25                          ; build Sector offset in Cluster
   ret
[end]



[_FileWriteByteArray]
$EXTERNAL _AddrFileBuffer2Z , _AddrFileHandleBase2Z , _CheckEOF , _CheckFileAccess_Mode
$EXTERNAL _ClearDOSError , _FileWriteByte , _LoadFilePositionYZ2A , _LoadFileSizeYZ2B
$EXTERNAL _SaveFilePositionA2YZ , _SaveFileSizeA2YZ , _SetBitFileStatusYZ , _SetFileWritePending
   ;---------------------------------------------------------------------------
   ; write a SRAM range to the file
   ; Input: Y+0,1 Filehandle
   ;        X Pointer to Character transfer to file
   ;        r20,21 : number of bytes to transfer
   ;---------------------------------------------------------------------------
_FileWriteByteArray:
 * ldi r25 , cFileWrite_Mode
```

```
   rcall _CheckFileAccess_Mode
   brcs _FileWriteByteArray90
_FileWriteByteArray0:
   rcall _FilePosition2A_BufferByte
_FileWriteByteArray1:
   subi r20, 1                          ; decr. byte counter
   sbci r21, 0
   brcs _FileWriteByteArray80           ; last byte written
   ld r24, X+                           ; next byte to transfer to file
 * sbrc r25 , dFileEmpty
   rjmp _FileWriteByteArray2
   cpi r16, &HFF                        ; at skip to next sector
   brne _FileWriteByteArray3
   sbrs r17, 0                          ; if bit8 = 1 than realy sector skip
   rjmp _FileWriteByteArray3            ; no sector skip
_FileWriteByteArray2:
   rcall _SaveFilePositionA2YZ          ; save file Position
   push xh
   push xl
   push r21
   push r20
   rcall _FileWriteByte0                ; do all handle with next Sector
   pop r20
   pop r21
   pop xl
   pop xh
   brcs _FileWriteByteArray90           ; Error ?
   rjmp _FileWriteByteArray0
_FileWriteByteArray3:
   st Z+, r24
   subi r16, &HFF                       ; +1
   sbci r17, &HFF
   rjmp _FileWriteByteArray1
_FileWriteByteArray80:
   rcall _SetFileWritePending           ; also loads Filehandle to Z
   rcall _LoadFileSizeYZ2B              ; File Size to B
   cp r20, r16
   cpc r21, r17
   cpc r22, r18
   cpc r23, r19
   brcc _FileWriteByteArray81           ; filesize to adjust?
   rcall _SaveFileSizeA2YZ              ; Position is new file-size
 * ldi r24 , dmFATDirWritePending       ; set flat for FAT / Dir write is pending
   rcall _SetBitFileStatusYZ
_FileWriteByteArray81:
   rcall _SaveFilePositionA2YZ
   rcall _CheckEOF
   rjmp _ClearDOSError
_FileWriteByteArray90:
   ret

   ; Load file position and build pointer to byte in buffer
_FilePosition2A_BufferByte:
   rcall _AddrFileHandleBase2Z
 * ldd r25 , Z + co_FileBufferStatus
   rcall _LoadFilePositionYZ2A
   rcall _AddrFileBuffer2Z              ; to buffer start
   mov r23, r17
   andi r23, &H01
   add zl, r16
   adc zh, r23
   ret

[end]
```

```
[_CheckEOF]
$EXTERNAL _AddrFileHandleBase2Z , _LoadFilePositionYZ2B , _LoadFileSizeYZ2A
$EXTERNAL _SUB32
   ;--------------------------------------------------------------------------------
   ; check file position for being at EOF
   ; it sets/clears bit EOF in file buffer status
   ; if EOF then C-Flag is set
   ;--------------------------------------------------------------------------------
_CheckEOF:
   ldd zl, Y+0
   ldd zh, Y+1
 * ldd r24 , Z + co_FileBufferStatus
   rcall _LoadFileSizeYZ2A               ; load file size to A
   rcall _LoadFilePositionYZ2B           ; load file position to B
   rcall _AddrFileHandleBase2Z
  call _Sub32
   brcs _CheckEOF3
   breq _CheckEOF3
 * cbr r24 , dmEOF
 * std Z + co_FileBufferStatus , r24
   clc
   ret
_CheckEOF3:
 * sbr r24 , dmEOFAll
 * std Z + co_FileBufferStatus , r24
   sec
   ret
[end]


[_SetDirTimeDate]
   ;--------------------------------------------------------------------------------
   ; sets actual time and date in DIR-Entry
   ; X-> Time entry in DIR Entry
   ; Entry in DIR is:
   ; Time:
   ; Bit 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
   ;     +--- Hour ---+ +--- Minute ----+  +-- Sec/2 --+
   ; Date:
   ; Bit 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
   ;     +---- Year+20 -----+  +-Month -+  +--- Day ---+   Years start ab 1980 with 0
   ;--------------------------------------------------------------------------------
_SetDirTimeDate:
*#IF VarExist("_Sec")
   *#IF _USERCLOCK = 1
     push xh
     push xl
     call Getdatetime                    ; THIS IS IN THE MAIN PROG
     pop xl
     pop xh
  *#ENDIF
   loadadr _Sec , Z
   ;_SetDirTimeDate_Z:
   ld r10, Z+                            ; Sec
   ld r11, Z+                            ; Min
   ld r12, Z+                            ; Hour
   ld r13, Z+                            ; Day
   ld r14, Z+                            ; Month
   ld r15, Z+                            ; Year
*#ELSE
   ldi r24, 1
   clr r10
   clr r11
   mov r12, r24
   mov r13, r24
   mov r14, r24
   mov r15, r24
*#ENDIF
   ;_SetDirTimeDate_Reg:
   lsl r12
   lsl r12
   lsl r12
   clr r24
```

```
   lsr r11
   ror r24
   lsr r11
   ror r24
   or r24, r10
   lsr r11
   ror r24
   or r11, r12
   st X+, r24                          ; Write time to DIR
   st X+, r11
   ldi r24, 20                         ; DOS Year start with 1980 so Year + 20 we are past 1999
   add r24, r15
   swap r14
   lsl r14
   rol r24
   or r14, r13
   st X+, r14
   st X+, r24
   ret
[End]


[_GetDiskFreeSize]
$EXTERNAL _GetDiskSize , _GetFreeCluster
   ;--------------------------------------------------------------------------------
   ; Get free size on disk/drive
   ; checks all cluster entries for being 00 00
   ; Free size = (free clusters) * (sectors per Cluster)  /2
   ; OUTPUT: Size in A (r16-r19)
   ;--------------------------------------------------------------------------------
_GetDiskFreeSize:
   rcall _GetFreeCluster0
   * cpi r25 , cpNoFreeCluster         ; GetFreeCluster0 should end with this Error
   brne _GetDiskFreeSize1
   mov r16, r20
   mov r17, r21
   mov r18, r22
   mov r19, r23
   rjmp _GetDiskSize0
_GetDiskFreeSize1:
   ret
[end]


[_GetDiskSize]
$EXTERNAL _ClearDOSError , _LongX2A
   ;--------------------------------------------------------------------------------
   ; Get Disk size in KBytes (1024 Bytes)
   ; Disk Size = ( (Max Cluster Number) -1 ) * ( sectors per cluster) / 2
   ; Result: Size in A (r16-r19) LONG
   ;--------------------------------------------------------------------------------
_GetDiskSize:
   loadadr glMaxClusterNumber , X
   rcall _LongX2A
   subi r16, 1                         ; -1
   sbci r17, 0
   sbci r18, 0
   sbci r19, 0
_GetDiskSize0:
   lds r24, {gbSectorsPerCluster}
   sbrc r24, 0                         ; 1 Sector per cluster; shift 1 bit to right
   rjmp _GetDiskSize2
_GetDiskSize1:
   sbrc r24, 1
   rjmp _GetDiskSize9
   lsl r16
   rol r17
   rol r18
   rol r19
   lsr r24
   rjmp _GetDiskSize1
_GetDiskSize2:
```

```
   lsr r19
   ror r18
   ror r17
   ror r16
_GetDiskSize9:
   rjmp _ClearDOSError
[End]
```

```
[_FilesAllFlush]
$EXTERNAL _ClearDOSError , _SaveFileSector , _SetBitFileStatusYZ
   ;--------------------------------------------------------------------------------
   ; Flushes all open files to disk/drive if necessary
   ;--------------------------------------------------------------------------------
_FilesAllFlush:
 * ldi r23 , cFileHandles                ; max. Number of file -handles
   loadadr abFileHandles(1) , Z
_FilesAllFlush1:
   ld r24, Z
   tst r24                               ; valid file# in filehandle
   breq _FilesAllFlush2
   st -Y, zh
   st -Y, zl
   push r23
 * ldi r24 , dmFATDirSaveAnyWay
   rcall _SetBitFileStatusYZ
   rcall _SaveFileSector
   pop r23
   ld zl, Y+
   ld zh, Y+
   brcs _FilesAllFlush4
_FilesAllFlush2:
   dec r23
   breq _FilesAllFlush3                  ; All filehandle searched
 * subi zl , lByte(c_FileHandleSize_M)   ; advance filehandle pointer to next block
 * sbci zh , hbyte(c_FileHandleSize_M)
   rjmp _FilesAllFlush1
_FilesAllFLush3:
   rjmp _ClearDOSError
_FilesAllFlush4:
   ret
[end]
```

```
[_CompareRange]
$EXTERNAL _Mem_Copy
   ; check a range of SRAM (register)
   ; make a cp Z, X
_CompareRange_Update:
   set                                   ; replace X with Z if Z < X
   rjmp _CompareRange1
_CompareRange:
   clt
_CompareRange1:
   st -Y, zh
   st -Y, zl
   st -Y, xh
   st -Y, xl
   mov r0, r24                           ; Save counter for replace
   cp r24, r24                           ; clear Carry and set Zero bit for next cpc
 * in r12 , SREG
_CompareRange2:
   ld r10, Z+
   ld r11, X+
 * out SREG , r12
   cpc r10, r11
 * in r12 , SREG
   dec r24
   brne _CompareRange2
```

```
 * out SREG , r12
   brtc _CompareRange9
   brcc _CompareRange9
   ; replace X with Z, if Z < X
   mov r24, r0
   ld xl, Y+0
   ld xh, Y+1
   ld zl, Y+2
   ld zh, Y+3
   rcall _Mem_Copy
 * out SREG , r12
_CompareRange9:
   ld xl, Y+
   ld xh, Y+
   ld zl, Y+
   ld zh, Y+
   ret
[end]
```

Labelindex:

_A2LongX 29
_A2LongY 6
_A2LongZ 29
_A2WordX 31
_Add32_Byte 28
_Add32_r 28
_Add32_Word 28
_AddrFileBuffer2X 36
_AddrFileBuffer2Z 37
_AddrFileBufferStatus2Z 34
_AddrFileFirstCluster2X 32
_AddrFileHandleBase2Z 32
_AddrFilePosition2X 36
_AddrFilePosition2Z 35
_AddrFileSectorNumber2X 37
_AddrFileSectorNumber2Z 34
_AddrFileSize2X 34
_AddrFileSize2Z 33
_AllocateFirstFileSector 61
_AllocateNextFileSector 60
_AVRDOSVer 1
_B2LongZ 30
_B2WordX 30
_B2WordZ 31
_BLoad 15
_BSave 16
_ChDir 8
_ChDir0 8
_Check_Root 8
_CheckA_Zero 31
_CheckAvailableSectorSize 59
_CheckBootSign 2
_CheckEOF 76
_CheckFileAccess_Mode 65
_CheckFileAccess_Mode_Z 65
_CheckFileInUse 60
_CheckFileSystemType 2
_CheckUpdateFreeCluster 59
_CheckUpdateFreeDirEntry 59
_ClearA 28
_ClearBitFileStatusYZ 39
_ClearDirCluster 63
_ClearDirEntry 57
_ClearDOSError 38
_ClearDOSError_add2toY 38
_ClearErrorFlag 38
_ClearErrorFlag_add2toY 38
_ClearFileHandle 53
_ClearFileHandle_Y 53
_ClearFileSector 63
_ClearFileSystem 2
_ClearSector 63
_CloseFileHandle 54
_CloseFileHandle_Y 54
_Cluster2Sector 25
_ClusterA2FAT 36
_Comp_String_Mem 48
_CompareRange 78
_CompareRange_Update 78
_CreateFile 56
_CreateFile_CheckedNotExists 56
_CreateFile_DeleteOld 56
_DeleteFile 57
_DeleteFile0 57
_DeleteFile0a 58
_DeleteFile0b 58
_Dir 20
_Dir0 20
_DriveReadSector_SEV 41
_DriveWriteSector_SEV 41
_FileAttr 17
_FileByteInput 19
_FileByteOutput 19
_FileClose 16
_FileDateS 22
_FileDateS0 22
_FileDateTimeB 21
_FileDateTimeB0 21
_FileDateTimeS 21
_FileDateTimeS0 22
_FileEOF 17
_FileFlush 19
_FileGetRange 18
_FileGetRange_Bytes 18
_FileLen 21
_FileLen0 21
_FileLineInput 14
_FileLineOutput 14
_FileLoc 13
_FileLOF 13
_FileOpen_Append 11
_FileOpen_Append0 11
_FileOpen_Binary 12
_FileOpen_Binary0 12
_FileOpen_FileNotExist 13
_FileOpen_Input 11
_FileOpen_Input0 10
_FileOpen_Output 12
_FileOpen_Output0 12
_FilePosition2A_BufferByte 75
_FilePutRange 17
_FilePutRange_Bytes 17
_FileReadByte 55
_FileReadByte0 55
_FileReadByteArray 73
_FileReadByteArray0 73
_FileReadDec2Num 72
_FileReadDecDouble 73
_FileReadDecSingle 72
_FileReadString 71
_FileReadString2StrBuf 71
_FileRename 10
_FilesAllFlush 78
_FileSeek 13
_FileSeekSet 14
_FileTimeS 23
_FileTimeS0 23
_FileWrite_StrBuf 71
_FileWriteByte 65
_FileWriteByte0 65
_FileWriteByteArray 74
_FileWriteByteArray0 75
_FileWriteComma 69
_FileWriteCRLF 69
_FileWriteDecByte 70
_FileWriteDecDouble 71

_FileWriteDecInt 70
_FileWriteDecLong 70
_FileWriteDecSingle 70
_FileWriteDecWord 70
_FileWriteQuotationMark 69
_FileWriteStrConst 69
_FileWriteString 68
_FileWriteStringQuotationMark 68
_GetAttr 23
_GetAttr0 23
_GetDirEntry 47
_GetDirEntry_0 47
_GetDirEntry_DirStart 47
_GetDirEntry_Y 47
_GetDirEntryFreeStart 47
_GetDiskFreeSize 77
_GetDiskSize 77
_GetDiskSize0 77
_GetFileAttribute 50
_GetFileDate 50
_GetFileExtend 50
_GetFileFirstCluster 50
_GetFileHandle 53
_GetFileName 49
_GetFileSize 51
_GetFileSystem 3
_GetFileSystemErrorExit 4
_GetFileTime 51
_GetFreeCluster 26
_GetFreeCluster0 26
_GetFreeDirEntry 46
_GetFreeFileNumber 53
_GetNextCluster 25
_GetNextCluster_Y 25
_GetNextDirSectorNumber 45
_GetNextSectorNumber 24
_GetSector 42
_GetSector_Dir 42
_GetSector_FAT 42
_GetSector_File 42
_GotoFilePosition 66
_GotoFilePosition_1Base 66
_GotoFilePositionEOF 67
_InitDirEntry 6
_InitDirEntrySub 6
_LastCluster2FAT 36
_LastCluster2FATT 36
_LoadFileBufferStatusYZ 34
_LoadFileFirstClusterYZ2A 33
_LoadFilePositionYZ2A 35
_LoadFilePositionYZ2B 35
_LoadFileSectorNumberYZ2A 34
_LoadFileSizeYZ2A 33
_LoadFileSizeYZ2B 34
_LoadNextFileSector 43
_LoadNextFileSector_Position 43
_LongX2A 28
_LongX2B 29
_LongY2A 6
_LongZ2A 29
_LongZ2B 29
_Mem_Copy 32
_MkDir 7
_Mov4A2B 31
_NormFileName 48
_NormFileName0 48
_OpenFile 54
_Position2SeqCluster 74
_ReadDirSector 41
_RMDir 9
_SaveA2LongY 6
_SaveFileBufferStatusYZ 35
_SaveFileDirSectorNumberA2YZ 33
_SaveFileFirstClusterA2YZ 33
_SaveFilePositionA2YZ 35
_SaveFilePositionB2YZ 35
_SaveFileSector 62
_SaveFileSectorNumberA2YZ 37
_SaveFileSizeA2YZ 33
_SaveSector_DIR 64
_SaveSector_FAT 64
_SaveSector2FATCopy 64
_SearchDirEntry 44
_SearchDirEntry_CheckFileName 44
_SearchNextDirEntry 48
_Sector2Cluster 24
_SetBitEOFYZ 39
_SetBitFileEmptyYZ 39
_SetBitFileStatusYZ 39
_SetDirTimeDate 76
_SetDirWritePending 40
_SetDOSError 38
_SetDOSError_add2toY 38
_SetErrorFlag 38
_SetErrorFlag_add2toY 38
_SetFATWritePending 39, 40
_SetFileHandle 51
_SetFileNameWildCardError 38
_SetFileNameWildCardError0 38
_SetFileWritePending 39
_ShiftL_WordB 32
_ShiftR_WordB 32
_Sub32_Byte 28
_Sub32_Word 28
_WordX2B 30
_WordX2B_L 30
_WordZ2A 30
_WordZ2A_L 30
_WordZ2B 30