fatfs.c

```c
void MX_FATFS_Init(void)
{
  /*## FatFS: Link the SD driver ############*/
  retSD = FATFS_LinkDriver(&SD_Driver, SDPath);
  /* additional user code for init */
}
```

ff_gen_drv.c

```c
uint8_t FATFS_LinkDriverEx(const Diskio_drvTypeDef *drv, char
*path, uint8_t lun)
{
  uint8_t ret = 1;
  uint8_t DiskNum = 0;

  if(disk.nbr < _VOLUMES)
  {
    disk.is_initialized[disk.nbr] = 0;
    disk.drv[disk.nbr] = drv;
    disk.lun[disk.nbr] = lun;
    DiskNum = disk.nbr++;
    path[0] = DiskNum + '0';
    path[1] = ':';
    path[2] = '/';
    path[3] = 0;
    ret = 0;
  }
  return ret;
}

uint8_t FATFS_LinkDriver(const Diskio_drvTypeDef *drv, char *path)
{
  return FATFS_LinkDriverEx(drv, path, 0);
}
```

ff_gen_drv.h

```c
typedef struct
{
  DSTATUS (*disk_initialize) (BYTE);
  DSTATUS (*disk_status)     (BYTE);
  DRESULT (*disk_read)       (BYTE, BYTE*, DWORD, UINT);
#if _USE_WRITE == 1
  DRESULT (*disk_write)      (BYTE, const BYTE*, DWORD, U
#endif /* _USE_WRITE == 1 */
#if _USE_IOCTL == 1
  DRESULT (*disk_ioctl)      (BYTE, BYTE, void*);
#endif /* _USE_IOCTL == 1 */
}Diskio_drvTypeDef;

typedef struct
{
  uint8_t                 is_initialized[_VOLUMES];
  const Diskio_drvTypeDef *drv[_VOLUMES];
  uint8_t                 lun[_VOLUMES];
  volatile uint8_t        nbr;
}Disk_drvTypeDef;
```

ff.c

```c
f_read(FIL* fp, void* buff, UINT btr, UINT* br)
{ ...
  disk_read(fs->drv, fp->buf, sect, 1)
}
```

diskio.c

```c
disk_read(BYTE pdrv, BYTE *buff, DWORD sector, UINT count)
{ ...
  disk.drv[pdrv]->disk_read(disk.lun[pdrv], buff, sector, count);
}
```

sd_diskio.c

```c
const Diskio_drvTypeDef  SD_Driver =
{
  SD_initialize,
  SD_status,
  SD_read,
#if  _USE_WRITE == 1
  SD_write,
#endif /* _USE_WRITE == 1 */

#if  _USE_IOCTL == 1
  SD_ioctl,
#endif /* _USE_IOCTL == 1 */
};

SD_read(BYTE lun, BYTE *buff, DWORD sector, UINT count)
{ ...
  BSP_SD_ReadBlocks((uint32_t*)buff, (uint32_t) (sector), count, SD_TIMEOUT)
}
```

bsp_driver_sd.c

```c
uint8_t BSP_SD_ReadBlocks(uint32_t *pData, uint32_t ReadAddr, uint32_t NumOfBlocks, uint32_t Timeout)
{ ...
  HAL_SD_ReadBlocks(&hsd, (uint8_t *)pData, ReadAddr, NumOfBlocks, Timeout}
}
```

Stm32f4xx_hal_sd.c

```c
HAL_SD_ReadBlocks(SD_HandleTypeDef *hsd, uint8_t *pData, uint32_t BlockAdd, uint32_t NumberOfBlocks, uint32_t Timeout)
{ ...
}
```