FUSION-C

New functions added to version 1.2 from the book Fusion-C version 1.1

PrintHex	V1.2	**CONSOLE**
void	**PrintHex (unsigned int num)**	
Prints the hexadecimal representation of the integer ***num***, on the text screen mode		

CheckBreak	V1.2	**CONSOLE**
int	**CheckBreak(void)**	
Checks the CTRL-BREAK in the MSX-DOS console. Return 0 if not pressed, or -1 if pressed		

PutCharHex	V1.2	**CONSOLE**
void	**PutCharHex(char c)**	
Prints the hexadecimal representation of the char ***num***, on the text screen mode		

MouseReadTo	V1.2	**MOUSE**

void MouseReadTo(unsigned char MousePort, MOUSE_DATA *md);

Reads and returns the mouse offsets, and the 2 buttons states of the mouse connected in MousePort.

The function is using this pre-defined structure

```
typedef struct {
  signed char dx;
  signed char dy;
  unsigned char lbutton;
  unsigned char rbutton;
} MOUSE_DATA;
```

You must declare this structure before using this function, like this :
static MOUSE_DATA mb;

MousePort must be 1 or 2 depending on the mouse port you want to read.

returned values goes to the structure variables. According to the previous structure declaration, you will receive data inside
mb.dx
mb.dy
mb.lbutton
mb.rbutton

lbutton and lbutton are set to 0 when pressed

Code example :
MouseReadTo(1,&mb)

StrReverse	V1.2	**STRING**
char*	**StrReverse(char *str)**	
This function reverse the order of the chars inside the string ****str*** The new string is returned as a string chars		

Itoa	V1.2	**STRING**
char*	**Itoa(int num, char* str, int base)**	
This function convert an the integer ***num*** to a string of chars ****str.*** The new string is return as a string of chars, and must be declared before using this function. ***base*** indicates which base you want to convert to : 8, 10, 16		

BoxFill	V1.2	**VDP_GRAPH2**
void BoxFill (int X1, int Y1, int X2, int yY22, char color, char OP)		
Draws a filled rectangle ***from X1,Y1*** (left upper corner) to ***x2,y2*** (right bottom corner) with ***color*** and logical operation **OP**.		

HMMM	V1.2	**VDP_GRAPH2**
void HMMM(int XS, int YS, int XT, int YT, int DX, int DY);		
High speed from VRAM to VRAM Copy the rectangle image starting at ***XS,YS*** *(top left corner of the rectangle)* to the target ***XT,YT*** coordonate. Length and high of the rectangle image are defined by ***DX*** and ***DY.*** No logical operation allowed.		

LMMM	V1.2	**VDP_GRAPH2**
void LMMM (int XS, int YS, int XT, int YT, int DX, int DY, unsigned char OP)		
High speed copy with logical Operation from VRAM to VRAM Copy the rectangle image starting at ***XS,YS*** *(top left corner of the rectangle)* to the target coordinates ***XT,YT***. Length and high of the rectangle image are defined by ***DX*** and ***DY*** ***OP*** must be a standard logical operator If you want to copy a rectangle image from one vram page to another, use the ***YT*** coordonate. For example, if YT>255 you are working on the 2nd VRAM page		

YMMM	V1.2	VDP_GRAPH2
void YMMM(int XS, int YS, int DY, int NY, int DiRX)		
High speed copy of a part of image from VRAM to VRAM . This only copy the image part to another Y position (***DY***) The rectangle image starting at ***XS,YS,*** and ends at ***255,YS+NY*** if ***DirX*** =0 or ends at ***0,YS+NY*** if ***DiRX***=1 The image block is copied to ***XS, DY*** position. No logical operation allowed.		

LMMC	V1.2	VDP_GRAPH2
void LMMC (void *pixeldatas, int X, int Y, int DX, int DY, unsigned char OP);		
Copy the RAM ****pixeldata*** buffer to Vram ***X,Y*** position with logical operation ***DX*** is Length of the zone to copy ***DY*** is Height of the zone to copy Use ***Y*** Coordinate > 256 to copy buffer on other page. (add 256 to Y to copy to page 1, in screen 8); ***OP*** is a standard logical operator parameter. *In Screen mode 5 or 7, if you want ot use LMMC command, you must previously tranfer data to RAM buffer with HMCM_SC8 instead of HMCM*		

HMMV	V1.2	**VDP_GRAPH2**

void HMMV(int XS, int YS, int DX, int DY, char COLOR)

High speed fillinglmmc of a rectangle box.
Rectangle top left corner is defined ***at XS,YS*** its length is ***DX*** pixels, its height is ***DY*** pixels. The color to use is defined by ***COLOR***
No logical operation allowed.
When working on screen 5 or 7, HMMV will fill 2 horizontal pixels at the same time. The ***COLOR*** variable must be divided into two blocks of 4 bits example ***COLOR l***: ***0bAAAA BBBB .*** The left 4 bits will be used for the left pixel color, and the 4 right bits will be used for right pixel.
If you do not need this feature, your ***COLOR*** variable can be calculated by this formula :

```
color =12;                      // use color 12
color=((color << 4) | color);   // Use color 12 for both pixels
```

LMMV	V1.2	**VDP_GRAPH2**

void LMMV(int XS, int YS, int DX, int DY, char COL, unsigned char OP)

High speed fill of a rectangle box, with logical operation.
Rectangle top left corner is defined ***at XS,YS*** its length is ***DX*** pixels, its height is ***DY*** pixels. The color to use is defined by ***COL,*** and the logical operation is ***OP.***