BUFFERCHIP.CP

```
#line 1 "C:/Users/Public/Documents/Mikroelektronika/Visual
TFT/Projects/NewDesignFT800/NewDesign_Code/mikroC PRO
for DSPIC/bufferchip.c"
#line 1 "c:/users/public/documents/mikroelektronika/visual
tft/projects/newdesignft800/newdesign_code/mikroc pro for
dspic/bufferchipheader.h"
sbit Mmc_Chip_Select at LATB6_bit;
sbit Mmc_Chip_Select_Direction at TRISB6_bit;

volatile unsigned char SDBuff[5120];
volatile unsigned int SDPTR=0;
unsigned char*
SDBuffPTRS[]={&SDBuff[0],&SDBuff[512],&SDBuff[1024],&S
DBuff[1536],

&SDBuff[2048],&SDBuff[2560],&SDBuff[3072],&SDBuff[3584]
,
 &SDBuff[4096],&SDBuff[4608]};

volatile unsigned long Currentsect=0;
volatile unsigned long SwapstartSector=0;

volatile unsigned int dataReadyflag1=0;
volatile unsigned int dataReadyflag2=0;
volatile unsigned int dataReadyflag3=0;
volatile unsigned int dataReadyflag4=0;
volatile unsigned int dataReadyflag5=0;
volatile unsigned int dataReadyflag6=0;
volatile unsigned int dataReadyflag7=0;
volatile unsigned int dataReadyflag8=0;
volatile unsigned int dataReadyflag9=0;
volatile unsigned int dataReadyflag10=0;
unsigned int*
dataReadyflags[]={&dataReadyflag1,&dataReadyflag2,&dataRead
yflag3,&dataReadyflag4,

&dataReadyflag5,&dataReadyflag6,&dataReadyflag7,&dataReady
flag8,
 &dataReadyflag9,&dataReadyflag10};


volatile unsigned int newFileflag=0;

volatile unsigned char newFileName[10];
volatile unsigned int newFileNamePtr=0;
volatile unsigned int nameStart=0;

volatile unsigned int stopLoggingFlag=0;
volatile short logtxt;
volatile short error;
volatile unsigned char errorString[5];

volatile unsigned int startListeningFlag=0;
#line 2 "C:/Users/Public/Documents/Mikroelektronika/Visual
TFT/Projects/NewDesignFT800/NewDesign_Code/mikroC PRO
for DSPIC/bufferchip.c"
void LEDIndicator(){
 LATA.B1=1;
 Delay_ms(1000);
 LATA.B1=0;
 Delay_ms(1000);
}

void mappins(){
 TRISB.B4=0;
 TRISA.B4=1;

 TRISB.B12=1;
 TRISB.B11=0;
 TRISB.B10=1;

 TRISA.B1 = 0;
 LATA.B1=1;

 PPS_Mapping(20,_INPUT,_U1RX);
 PPS_Mapping(36,_OUTPUT,_U1TX);

 PPS_Mapping(44,_INPUT,_SDI2IN);
 PPS_Mapping(43,_OUTPUT,_SDO2);
 PPS_Mapping(42,_INPUT,_SCK2IN);

}
void initSPICommunication(){
 SPI2_Init_Advanced(_SPI_SLAVE,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_2,
 _SPI_PRESCALE_PRI_4,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_MIDDLE,
 _SPI_CLK_IDLE_LOW,
 _SPI_ACTIVE_2_IDLE);
 IFS2.B1=0;
 IPC8.B6=1;
 IPC8.B5=1;
 IPC8.B4=0;
 SPI2STAT.B4=0;
 SPI2STAT.B3=0;
 SPI2STAT.B2=1;
 IEC2.B1=1;

}
void Spiinterrupt() iv IVT_ADDR_SPI2INTERRUPT{
 unsigned tem;
 IFS2.B1=0;
 tem=SPI2BUF;


 if(startListeningFlag){

 SDbuff[SDPTR++]=tem;
 if(SDPTR==510){SDPTR=512;dataReadyflag1=1;}
 if(SDPTR==1022){SDPTR=1024;dataReadyflag2=1;}
 if(SDPTR==1534){SDPTR=1536;dataReadyflag3=1;}
 if(SDPTR==2046){SDPTR=2048;dataReadyflag4=1;}
 if(SDPTR==2558){SDPTR=2560;dataReadyflag5=1;}
 if(SDPTR==3070){SDPTR=3072;dataReadyflag6=1;}
 if(SDPTR==3582){SDPTR=3584;dataReadyflag7=1;}
 if(SDPTR==4094){SDPTR=4096;dataReadyflag8=1;}
 if(SDPTR==4606){SDPTR=4608;dataReadyflag9=1;}
 if(SDPTR==5118){SDPTR=0;dataReadyflag10=1;}
 }

}
void initUARTCommunication(){

UART1_Init_Advanced(921600,_UART_8BIT_NOPARITY,_UA
RT_ONE_STOPBIT,_UART_HI_SPEED);
 Delay_ms(200);
 IFS0.B11=0;
 U1STA.OERR=0;
 IPC2.B14=1;
 IPC2.B13=0;
 IPC2.B12=0;
 IEC0.B11=1;
}
void U1interrupt() iv IVT_ADDR_U1RXINTERRUPT{
 unsigned char temp;
 IFS0.B11=0;
```

```
 temp=U1RXREG;

 if(nameStart){
 newFileName[newFileNamePtr++]=temp;
 if(temp==0x00){
 newfileFlag=1;
 newFileNamePtr=0;
 nameStart=0;
 }
 if(newFileNamePtr==10){newFileNamePtr=0;}
 }
 else
 {
 if(temp==0xF3){
 nameStart=1;
 }
 if(temp==0xE2){
 stopLoggingFlag=1;
 }
 if(temp==0xD3){
 startListeningFlag=1;
 SDPTR=0;
 }

 }


}
void initSD(){
 unsigned int i=0;
 SPI1_Init_Advanced(_SPI_MASTER,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_2,
 _SPI_PRESCALE_PRI_16,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_END,
 _SPI_CLK_IDLE_HIGH,
 _SPI_ACTIVE_2_IDLE) ;

 while(FAT32_Init()!=0){LEDIndicator();}

 SPI1_Init_Advanced(_SPI_MASTER,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_2,
 _SPI_PRESCALE_PRI_1,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_END,
 _SPI_CLK_IDLE_HIGH,
 _SPI_ACTIVE_2_IDLE);

 for(i=0;i<512;i++){SDBuff[i]=0xDD;}
 SPI_Set_Active(SPI1_Read, SPI1_Write);

}
void initAll(){
 PLLFBD = 68;
 CLKDIV = 0x0000;

 ANSELA = 0x00;
 ANSELB = 0x00;
 mappins();
 Delay_ms(500);
 initUARTCommunication();
 initSPICommunication();

 initSD();
}
void main() {
 unsigned int tempi=0;
 short log;
 unsigned char info[15];
 initAll();
 Delay_ms(500);
 LATA.B1=0;
 while(1){
 if(newfileFlag){
 newfileFlag=0;
 LATA.B1=1;

if(FAT32_MakeSwap(newFileName,30760,&SwapstartSector)==-
1){
 LATA.B1=1;
 while(1);
 }
 Currentsect=SwapstartSector;
 if(Currentsect==0){
 log=FAT32_Open("log", 0x04);
 LongWordToStr(Currentsect,info);
 FAT32_Write(log, "sect is zero", 13);
 FAT32_Close(log);
 }
 SDPTR=0;
 dataReadyflag1=0;
 dataReadyflag2=0;
 LATA.B1=0;
 UART1_Write(0xE7);
 }
 for(tempi=0;tempi<10;tempi++){
 if(*dataReadyflags[tempi]){
 *dataReadyflags[tempi]=0;
 LATA.B1=~LATA.B1;

if(FAT32_Dev_Write_Sector(Currentsect++,SDBuffPTRS[tempi])
!=0){
 LATA.B1=1;
 while(1);
 }
 if((Currentsect-SwapstartSector)==3075){stopLoggingFlag=1;}
 }
 }
 if(stopLoggingFlag){
 UART1_Write(0xF3);
 LATA.B1=1;
 Delay_ms(3000);
 LATA.B1=0;
 for(tempi=0;tempi<512;tempi++){
 SDbuff[tempi]=0x00;
 }
 if(FAT32_Dev_Write_Sector(Currentsect++,SDBuff)!=0){
 LATA.B1=1;
 while(1);
 }
 SDPTR=0;
 dataReadyflag1=0;
 dataReadyflag2=0;
 stopLoggingFlag=0;
 startListeningFlag=0;
 UART1_Write(0xA9);
 }
 }

 }
```

GPSCAMCHIP.CP

```
#line 1 "C:/Users/Public/Documents/Mikroelektronika/Visual
TFT/Projects/NewDesignFT800/NewDesign_Code/mikroC PRO
for DSPIC/GPSCamChip.c"
#line 1 "c:/users/public/documents/mikroelektronika/visual
tft/projects/newdesignft800/newdesign_code/mikroc pro for
dspic/camfunctionheader.h"
sbit Mmc_Chip_Select at LATB6_bit;
sbit Mmc_Chip_Select_Direction at TRISB6_bit;



unsigned char setCameraBaud[]={0x56, 0x00, 0x24, 0x03, 0x01,
0x1C ,0x4C};
unsigned char setImageSize[]={0x56 ,0x00, 0x54 ,0x01 ,0x00};
unsigned char takeAPic[]={0x56 , 0x00 , 0x36 , 0x01 , 0x00};
unsigned char readPicSize[]={ 0x56, 0x00, 0x34, 0x01, 0x00 };
unsigned char readPic[]={ 0x56, 0x00, 0x32, 0x0C, 0x00, 0x0A,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0x00, 0x0A };
unsigned char reset[]={ 0x56, 0x00, 0x26, 0x00 };
unsigned char stop[]={ 0x56, 0x00, 0x36, 0x01, 0x03 };

volatile unsigned char tempCam1=0;
volatile unsigned char tempCam2=0;
volatile unsigned int picSize=0;
unsigned char picStart=0;
volatile unsigned int EOF=0;
volatile unsigned int takePIC=0;
volatile unsigned int CameraBuffPointer=0;
volatile unsigned int
picbufferfull1=0,picbufferfull2=0,picbufferfull3=0,picbufferfull4=
0

,picbufferfull5=0,picbufferfull6=0,picbufferfull7=0,picbufferfull8=
0
 ,picbufferfull9=0,picbufferfull10=0;

volatile unsigned char CameraBuff[5120];


unsigned char*
buffersPtrs[]={&CameraBuff,&CameraBuff[512],&CameraBuff[1
024],&CameraBuff[1536],

&CameraBuff[2048],&CameraBuff[2560],&CameraBuff[3072],&
CameraBuff[3584],
 &CameraBuff[4096],&CameraBuff[4608]};
unsigned int*
bufferFullFlags[]={&picbufferfull1,&picbufferfull2,&picbufferfull
3,&picbufferfull4
 ,&picbufferfull5,&picbufferfull6,&picbufferfull7,&picbufferfull8
 ,&picbufferfull9,&picbufferfull10};

volatile unsigned char bufferLastFullSector;

unsigned long SDstartSector;
unsigned long SDcurrentSector;



volatile unsigned char GPSDelay=10;
volatile unsigned char GPSpackageReady=0;
volatile unsigned char GPSData[512];
volatile unsigned char GPSDataPtr=0;
unsigned char GPSintervalCounter=0;
unsigned char commandmss[]=
{0xA0,0xA1,0x00,0x09,0x08,0x01,0x00,0x00,0x00,0x00,0x00,0x
00,0x00,0x09,0x0D,0x0A};

unsigned char
GPSBaudcmd[]={0xA0,0xA1,0x00,0x04,0x05,0x00,0x05,0x00,0x
00,0x0D,0x0A};
unsigned char
GPSPowerMode[]={0xA0,0xA1,0x00,0x03,0x0C,0x00,0x00,0x0C
,0x0D,0x0A};
unsigned char
GPSUpdataRate[]={0xA0,0xA1,0x00,0x03,0x0E,0x05,0x00,0x0B,
0x0D,0x0A};

unsigned char
GPSWAASEnable[]={0xA0,0xA1,0x00,0x03,0x37,0x01,0x00,0x3
6,0x0D,0x0A};



unsigned int logGPS=0;
unsigned int GPSSecond=0;
unsigned int GPSMinute=0;
unsigned int GPSHour=0;
float GPSLatitude=0;
float GPSLongitude=0;
unsigned char GPSquality=0;
unsigned char tempLatitudeString[15];
unsigned char tempLongitudeString[15];
static unsigned char tempHourString[7];
static unsigned char tempMinuteString[7];
static unsigned char tempSecondString[7];
volatile unsigned int GPStoMasterPtr=0;
unsigned char* packageToSend;
unsigned char GPSString[70];
volatile unsigned int GPSStartFlag=0;
volatile unsigned int packageSize=0;


volatile unsigned int loggingmode=0;
volatile unsigned int startName=0;
volatile unsigned int samplingmode=0;


volatile unsigned char nameReceived[10];
volatile unsigned int nameReceivedPtr=7;
volatile unsigned int PicnumIncrementor=0;
volatile unsigned int newDir=0;

void processGPS();
unsigned char* processGPSString();
#line 2 "C:/Users/Public/Documents/Mikroelektronika/Visual
TFT/Projects/NewDesignFT800/NewDesign_Code/mikroC PRO
for DSPIC/GPSCamChip.c"
void LEDIndicator(){
 LATA.B1=0;
 Delay_ms(1000);
 LATA.B1=1;
 Delay_ms(1000);
 LATA.B1=0;
}
void mappins(){
 TRISA.B1=0;
 LATA.B1=1;
 TRISB.B0=1;
 TRISB.B5=0;
 TRISB.B1=1;
 TRISA.B4=0;
```

```
 TRISB.B12=1;
 TRISB.B11=0;
 TRISB.B10=1;

 PPS_Mapping(32,_INPUT,_U1RX);
 PPS_Mapping(37,_OUTPUT,_U1TX);
 PPS_Mapping(33,_INPUT,_U2RX);
 PPS_Mapping(20,_OUTPUT,_U2TX);

 PPS_Mapping(44,_INPUT,_SDI2IN);
 PPS_Mapping(43,_OUTPUT,_SDO2);
 PPS_Mapping(42,_INPUT,_SCK2IN);
}
void initSD(){
 unsigned int i=0;
 SPI1_Init_Advanced(_SPI_MASTER,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_2,
 _SPI_PRESCALE_PRI_16,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_END,
 _SPI_CLK_IDLE_HIGH,
 _SPI_ACTIVE_2_IDLE) ;

 while(FAT32_Init()!=0){LEDIndicator();}

 SPI1_Init_Advanced(_SPI_MASTER,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_2,
 _SPI_PRESCALE_PRI_1,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_END,
 _SPI_CLK_IDLE_HIGH,
 _SPI_ACTIVE_2_IDLE);

 for(i=0;i<5120;i++){CameraBuff[i]=0x00;}

}
void initSPI2CommunicationTOMaster(){
 SPI2_Init_Advanced(_SPI_SLAVE,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_4,
 _SPI_PRESCALE_PRI_4,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_MIDDLE,
 _SPI_CLK_IDLE_LOW,
 _SPI_ACTIVE_2_IDLE);


 SPI2STAT.B0=1;
 SPI2STAT.B1=0;
 SPI2STAT.B2=0;

 IFS2.B1=0;
 IPC8.B6=1;
 IPC8.B5=0;
 IPC8.B4=0;
 IEC2.B1=1;
}
void SpI2interrupt() iv IVT_ADDR_SPI2INTERRUPT{
 unsigned char temp;
 IFS2.B1=0;
 SPI2STAT.SPIROV=0;
 temp=SPI2BUF;

 if(!startName){
 switch(temp){
 case 0x31:
 loggingmode=0;
 samplingmode=1;


 temp=GPSString[GPSToMasterPtr++];
 SPI2BUF=temp;
 if(temp==0x00){GPStoMasterPtr=0;}
 break;
 case 0x22:
 samplingmode=0;
 GPStoMasterPtr=0;
 SPI2BUF=0x00;
 break;

 case 0x43:
 loggingmode=1;
 newDir=1;
 PicnumIncrementor=0;
 break;
 case 0x62:
 loggingmode=0;
 break;


 case 0x53:
 startName=1;

 nameReceivedPtr=0;
 default:break;
 }
 }
 else
 {
 if(temp==0xE4){
 startName=0;

 nameReceivedPtr=0;
 return;
 }
 nameReceived[nameReceivedPtr++]=temp;
 if(nameReceivedPtr==8){nameReceivedPtr=0;}
 }
}

void initCamera(){
 unsigned int iterator=0;
 UART1_Init(38400);
 Delay_ms(200);
 Delay_ms(2000);
 for(iterator=0;iterator<5;iterator++){
 UART1_Write(setImageSize[iterator]);
 }

 Delay_ms(150);

 for(iterator=0;iterator<7;iterator++){
 UART1_Write(setCameraBaud[iterator]);
 }
 Delay_ms(100);
 UART1_Init(57600);
 Delay_ms(200);
 U1STA.OERR=0;
}
void CameraInterrupt() iv IVT_ADDR_U1RXINTERRUPT{
 IFS0.B11=0;

 if(picStart==1)
 {
 tempCam2=UART1_Read();
 if(tempCam2==0xD9&&tempCam1==0xFF){
 CameraBuff[CameraBuffPointer++]=tempCam2;
```

```
 IEC0.B11=0;
 EOF=1;
 return;
 }

 CameraBuff[CameraBuffPointer++]=tempCam2;

if(CameraBuffPointer==512){picbufferfull1=1;bufferLastFullSect
or=0;}

if(CameraBuffPointer==1024){picbufferfull2=1;bufferLastFullSec
tor=1;}

if(CameraBuffPointer==1536){picbufferfull3=1;bufferLastFullSec
tor=2;}

if(CameraBuffPointer==2048){picbufferfull4=1;bufferLastFullSec
tor=3;}

if(CameraBuffPointer==2560){picbufferfull5=1;bufferLastFullSec
tor=4;}

if(CameraBuffPointer==3072){picbufferfull6=1;bufferLastFullSec
tor=5;}

if(CameraBuffPointer==3584){picbufferfull7=1;bufferLastFullSec
tor=6;}

if(CameraBuffPointer==4096){picbufferfull8=1;bufferLastFullSec
tor=7;}

if(CameraBuffPointer==4608){picbufferfull9=1;bufferLastFullSec
tor=8;}

if(CameraBuffPointer==5120){CameraBuffPointer=0;picbufferfull
10=1;bufferLastFullSector=9;}
 tempCam1=tempCam2;



 }
 else if(picStart==0)
 {

 CameraBuff[CameraBuffPointer]=UART1_Read();
 CameraBuffPointer++;
 if(CameraBuffPointer==14){picStart=1;CameraBuffPointer=0;}
 }

}
void takeOnePic(){
 unsigned int iterator=0;
 unsigned int sectorCount=0;
 unsigned int sectNumber=0;
 unsigned char PICNAMEString[10];
 short error=0;
 static unsigned int picnum=0;
 unsigned char picnumString[6];
 short int log=0;


for(iterator=0;iterator<5120;iterator++){Camerabuff[iterator]=0x0
0;}
 CameraBuffPointer=0;
 for(iterator=0;iterator<sizeof(takeAPic);iterator++){
 UART1_Write(takeAPic[iterator]);}
 Delay_ms(200);
 U1STA.OERR=0;
 U1STA.OERR=0;
 IFS0.B11=0;
 IPC2.B14=1;
 IPC2.B13=1;
 IPC2.B12=0;
 IEC0.B11=1;

 Delay_ms(100);

 CameraBuffPointer=0;

 for(iterator=0;iterator<sizeof(readPicSize);iterator++){
 UART1_Write(readPicSize[iterator]);
 }
 Delay_ms(15);
#line 227 "C:/Users/Public/Documents/Mikroelektronika/Visual
TFT/Projects/NewDesignFT800/NewDesign_Code/mikroC PRO
for DSPIC/GPSCamChip.c"
 picSize=CameraBuff[7];
 picSize=(picSize<<8)|CameraBuff[8];
 readPic[12]=CameraBuff[7];
 readPic[13]=CameraBuff[8];
 Delay_ms(15);

 if(picSize>70000||picsize<40000)
 {
 for(iterator=0;iterator<4;iterator++)
 {
 UART1_Write(reset[iterator]);
 }
 Delay_ms(100);
 InitCamera();
 takeOnePic();
 return;
 }

 sectNumber=picSize/512;
 if(picSize%512!=0){sectNumber++;}

 WordToStrWithZeros(PicnumIncrementor,picnumString);
 PICNAMEString[0]='P';
 PICNAMEString[1]=0x00;
 strcat(PICNAMEString,picnumString);
 LTRIM(PICNAMEString);
 RTRIM(PICNAMEString);
 StrCat(PICNAMEString,".jpg");


error=FAT32_MakeSwap(PICNAMEString,sectNumber,&SDstart
Sector);

 SDcurrentSector=SDstartSector;



 for(iterator=0;iterator<sizeof(readPic);iterator++){
 UART1_Write(readPic[iterator]);
 }



 while(1){
 for(iterator=0;iterator<10;iterator++){
 if(*(bufferFullFlags[iterator])){
 *(bufferFullFlags[iterator])=0;

FAT32_Dev_Write_Sector(SDcurrentSector++,buffersPtrs[iterator
]);
 sectorCount++;
 }
 }
```

```
 if(EOF)
 {
 bufferLastFullSector++;
 if(bufferLastFullSector==9){bufferLastFullSector=0;}

FAT32_Dev_Write_Sector(SDcurrentSector++,buffersPtrs[buffer
LastFullSector]);

 for(iterator=0;iterator<sizeof(stop);iterator++)
 {
 UART1_Write(stop[iterator]);
 }
 picStart=0;

 for(iterator=0;iterator<10;iterator++){
 *bufferFullFlags[iterator]=0;
 }
 CameraBuffPointer=0;

 tempCam1=0x00;
 tempCam2=0x00;
 EOF=0;


 break;
 }

 }

}

void initGPS(){

 unsigned char i=0;

 for(i=0;i<90;i++){GPSData[i]=0x00;}

 UART2_Init(9600);
 Delay_ms(500);

 for(i=0;i<11;i++){UART2_Write(GPSBaudcmd[i]);}
 Delay_ms(200);
 UART2_Init(115200);
 Delay_ms(200);
 for(i=0;i<10;i++){UART2_Write(GPSPowerMode[i]);}
 Delay_ms(200);

 for(i=0;i<16;i++){UART2_Write(commandmss[i]);}
 Delay_ms(200);

 for(i=0;i<10;i++){UART2_Write(GPSUpdataRate[i]);}
 Delay_ms(200);

 for(i=0;i<10;i++){UART2_Write(GPSWAASEnable[i]);}
 Delay_ms(200);

 IFS1.B14=0;
 U2STA.OERR=0;
 IPC7.B10=1;
 IPC7.B9=0;
 IPC7.B8=0;
 IEC1.B14=1;
}
void GPSinterrupt() iv IVT_ADDR_U2RXINTERRUPT {
 unsigned char temp=0;
 IFS1.B14=0;
 temp=UART2_Read();

if(temp==0x24){GPSDataPtr=0;GPSData[GPSDataPtr++]=temp;r
eturn;}
 GPSData[GPSDataPtr++]=temp;
 if(temp==0x2A){GPSpackageReady=1;processGPS();}
}
void processGPS(){
 unsigned char tempPtr=7;
 GPSHour=(GPSData[tempPtr]-48)*10+GPSData[tempPtr+1]-48;
 GPSMinute= (GPSData[tempPtr+2]-
48)*10+GPSData[tempPtr+3]-48;
 GPSSecond=(GPSData[tempPtr+4]-
48)*10+GPSData[tempPtr+5]-48;
 while(GPSData[tempPtr]!=','){tempPtr++;}
 tempPtr++;
 GPSLatitude=(GPSData[tempPtr]-48)*10+(GPSData[tempPtr+1]-
48)+
 (
 (GPSData[tempPtr+2]-48)*10+(GPSData[tempPtr+3]-48)+
 (GPSData[tempPtr+5]-48)*0.1+
 (GPSData[tempPtr+6]-48)*0.01+
 (GPSData[tempPtr+7]-48)*0.001+
 (GPSData[tempPtr+8]-48)*0.0001
 )/60;
 tempPtr+=10;
 if(GPSData[tempPtr]=='S'){GPSLatitude=-GPSLatitude;}
 tempPtr+=2;

 GPSLongitude=(GPSData[tempPtr]-
48)*100+(GPSData[tempPtr+1]-48)*10+(GPSData[tempPtr+2]-
48)+
 (
 (GPSData[tempPtr+3]-48)*10+
 (GPSData[tempPtr+4]-48)+
 (GPSData[tempPtr+6]-48)*0.1+
 (GPSData[tempPtr+7]-48)*0.01+
 (GPSData[tempPtr+8]-48)*0.001+
 (GPSData[tempPtr+9]-48)*0.0001
 )/60;
 tempPtr+=11;
 if(GPSData[tempPtr]=='W'){GPSLongitude=-GPSLongitude;}

}

unsigned char* processGPSString(){
 unsigned char ProcessedGPSString[70]="";
 WordToStr(GPSHour,tempHourString);
 WordToStr(GPSMinute,tempMinuteString);
 WordToStr(GPSSecond,tempSecondString);;
 FloattoStr(GPSLatitude,Ltrim(tempLatitudeString));
 FloattoStr(GPSLongitude,Ltrim(tempLongitudeString));
 strcat(ProcessedGPSString,Ltrim(tempHourString) );
 strcat(ProcessedGPSString,":" );
 strcat(ProcessedGPSString,Ltrim(tempMinuteString) );
 strcat(ProcessedGPSString,":" );
 strcat(ProcessedGPSString,Ltrim(tempSecondString) );
 strcat(ProcessedGPSString,"  *" );
```

```
 strcat(ProcessedGPSString,Ltrim(tempLatitudeString) );
 strcat(ProcessedGPSString,"," );
 strcat(ProcessedGPSString,Ltrim(tempLongitudeString) );

 packageSize=strlen(ProcessedGPSString)+1;
 memcpy(GPSString,ProcessedGPSString,70);
 return ProcessedGPSString;
}

void logGPSToSD(){
 short GPSfile;short log;
 short error;
 unsigned char errorString[6];
 unsigned char ProcessedGPSString[70];
 unsigned char num[6];

 memcpy(ProcessedGPSString,processGPSString(),70);
 strcat(ProcessedGPSString,",0 P" );
 WordToStrWithZeros(PicnumIncrementor,num);
 strcat(ProcessedGPSString,num);

 strcat(ProcessedGPSString,"\r\n" );

 GPSfile=FAT32_Open("GPS1.txt",0x04);
 FAT32_Write(GPSfile, GPSData, 68);
 FAT32_Write(GPSfile, "\r\n",2);

 FAT32_Close(GPSfile);

 GPSfile=FAT32_Open("GPS2.txt",0x04);
 FAT32_Write(GPSfile, ProcessedGPSString,
strlen(ProcessedGPSString));
 FAT32_Close(GPSfile);
 return;


 if(FAT32_Write(GPSfile, ProcessedGPSString,
Strlen(ProcessedGPSString))==-1){

 error = FAT32_GetError();
 ShortToStr(error,errorString);
 log=FAT32_Open("log", 0x04);
 FAT32_Write(log, errorString, strlen(errorString));
 FAT32_Close(log);while(1);

 }
 FAT32_Close(GPSfile);


}
void initAll(){
 PLLFBD = 68;
 CLKDIV = 0x0000;
 ANSELA = 0x00;
 ANSELB = 0x00;

 mappins();
 initSPI2CommunicationTOMaster();
 InitSD();
 initGPS();
 InitCamera();
}


void main() {
 short log;
 short error;
 unsigned char errorString[6];

 initAll();

 while(!GPSpackageReady);

 processGPSString();
 SPI2BUF=GPSString[0];
 GPStoMasterPtr=1;

 LATA.B1=0;
 while(1){
 while(samplingmode){
 if(GPSpackageReady){
 LATA.B1=1;
 GPSpackageReady=0;

 processGPSString();
 LATA.B1=0;




 Delay_ms(300);

 }


 }
 while(loggingmode){

 if(newDir){
 FAT32_ChangeDir("\\");
 Ltrim(nameReceived);
 Rtrim(nameReceived);
 if(FAT32_MakeDir(nameReceived)==-1){
 error = FAT32_GetError();
 ShortToStr(error,errorString);
 log=FAT32_Open("log", 0x04);
 FAT32_Write(log, errorString, strlen(errorString));
 FAT32_Close(log);
 while(1){
 LATA.B1=1;Delay_ms(100);LATA.B1=0;Delay_ms(100);
 }
 }
 FAT32_Changedir(nameReceived);
 newDir=0;
 }
 if(GPSpackageReady){
 LATA.B1=1;
 GPSpackageReady=0;

 PicnumIncrementor++;

 LogGPSToSD();
 LATA.B1=0;
 Delay_ms(100);
 LATA.B1=1;
 Delay_ms(100);
 LATA.B1=0;
 Delay_ms(100);
 LATA.B1=1;
 takeOnePic();
 LATA.B1=0;
 Delay_ms(500);
 }
 }



 }
}
```

NEWDESIGN_EVENTS_CODE.CP

```
#line 1
"C:/Users/ipm4/Desktop/latest_code_01062014/0121/NewDesignF
T800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/NewDesign_events_code.c"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_objects.h"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
#line 21
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
typedef unsigned char uint8_ft8;
typedef signed char int8_ft8;
typedef unsigned int uint16_ft8;
typedef int int16_ft8;
typedef unsigned long uint32_ft8;
typedef long int32_ft8;
typedef unsigned long long uint64_ft8;
typedef long long int64_ft8;

typedef enum {cmdREAD, cmdWRITE} TCmd;

typedef enum {opNONE, opREAD, opWRITE} TOpr;

typedef enum {umNONE, umDL=1<<0, umCP=1<<1,
umGRAM=1<<2} TUpdateMode;

typedef uint8_ft8 TFT800Alpha;
typedef uint8_ft8 TFT800Handle;

typedef struct
{
 int16_ft8 X;
 int16_ft8 Y;
} TFT800Point;

typedef struct
{
 int16_ft8 Left;
 int16_ft8 Top;
 uint16_ft8 Width;
 uint16_ft8 Height;
} TFT800Rect;

typedef union
{
 struct
 {
 uint8_ft8 B;
 uint8_ft8 G;
 uint8_ft8 R;
 };
 uint32_ft8 RGB;
} TFT800Color;

typedef struct
{
 TFT800Color Color;
 TFT800Alpha Alpha;
 uint8_ft8 Width;
} TFT800Pen;

typedef struct
{
 TFT800Color ColorBg;
 TFT800Color Color;


 TFT800Color ColorTo;
 uint8_ft8 Gradient;
 TFT800Alpha Alpha;
 uint8_ft8 Style;
} TFT800Brush;



typedef struct
{

 uint8_ft8 FontWidth[ 256 ];
 uint32_ft8 FontBitmapFormat;
 uint32_ft8 FontLineStride;
 uint32_ft8 FontWidthInPixels;
 uint32_ft8 FontHeightInPixels;
 uint32_ft8 PointerToFontGraphicsData;
} TFT800FontGraphics;

typedef struct
{
 TFT800FontGraphics GrData;
 uint16_ft8 FirstChar;
 uint16_ft8 LastChar;
} TFT800CharSetDsc;

typedef struct
{
 uint32_ft8 Name;
 uint32_ft8 Source;
 TFT800Color Color;
 TFT800Alpha Alpha;
 TFT800Handle Handle;
 TFT800CharSetDsc CharSetDsc;
} TFT800Font;

typedef struct
{
 uint8_ft8 Format;
 uint16_ft8 LineStride;
 uint16_ft8 Height;
} TFT800BmpLayout;

typedef struct
{
 uint8_ft8 FwFlags;
 uint16_ft8 Width;
 uint16_ft8 Height;
} TFT800BmpSize;

typedef struct
{
 uint32_ft8 Source;
 TFT800BmpLayout Layout;
 TFT800BmpSize Size;
} TFT800BmpConfig;

typedef struct
{
 TFT800BmpConfig Configs[16];
 uint8_ft8 Current;
} TFT800BmpHandle;

typedef struct
```

```
{
 TFT800BmpConfig Config;
 TFT800Color BlendColor;
 TFT800Alpha Alpha;
 TFT800Handle Handle;
} TFT800Bitmap;

typedef struct
{
 uint16_ft8 Hour;
 uint16_ft8 Min;
 uint16_ft8 Sec;
 uint16_ft8 mSec;
} TFT800Time;

typedef struct
{
 uint32_ft8 DrawingOptions;
 uint16_ft8 Val;
 uint16_ft8 Size;
 uint16_ft8 Range;
 uint16_ft8 Minor;
 uint16_ft8 Major;
 TFT800Time Time;
 uint8_ft8 Style;
} TFT800CoProcGraphics;

typedef struct
{
 TFT800Pen Pen;
 TFT800Brush Brush;
 TFT800Font Font;
 TFT800Bitmap Bitmap;

 TFT800Rect ClipRect;
 TFT800Point Cursor;

 uint8_ft8 Stencil;
 uint8_ft8 Tag;

 TFT800CoProcGraphics CPGraphics;
} TFT800Canvas;

typedef struct
{
 TOpr Opr;
 uint32_ft8 RWPtr;
} TFT800IO;


typedef struct
{

 uint32_ft8 Frequency;
 uint32_ft8 OutRenderMode;
 uint32_ft8 RenderReadScanLine;
 uint32_ft8 RenderWriteTrigger;

 uint32_ft8 hCycle;
 uint32_ft8 hOffset;
 uint32_ft8 hSize;
 uint32_ft8 hSync0;
 uint32_ft8 hSync1;

 uint32_ft8 vCycle;
 uint32_ft8 vOffset;
 uint32_ft8 vSize;
 uint32_ft8 vSync0;
 uint32_ft8 vSync1;
 uint32_ft8 Rotate180;
 uint32_ft8 OutBits;
 uint32_ft8 OutDither;
 uint32_ft8 OutSwizzle;
 uint32_ft8 OutCSpread;
 uint32_ft8 PClockPolarity;
 uint32_ft8 PClock;
} TFT800Display;

typedef struct
{
 uint32_ft8 TouchMode;
 uint32_ft8 TouchADCMode;
 uint32_ft8 TouchCharge;
 uint32_ft8 TouchSettle;
 uint32_ft8 TouchOversample;
 uint32_ft8 TouchRZThreshold;
} TFT800Touch;

typedef struct
{
 uint32_ft8 TransformA;
 uint32_ft8 TransformB;
 uint32_ft8 TransformC;
 uint32_ft8 TransformD;
 uint32_ft8 TransformE;
 uint32_ft8 TransformF;
} TFT800TouchTransform;

typedef struct
{
 uint8_ft8 Enable;
 uint8_ft8 Mask;
 uint8_ft8 Flags;
} TFT800Interrupt;

typedef struct
{
 uint8_ft8 Effect;
 uint8_ft8 Pitch;
 uint8_ft8 Volume;
 uint8_ft8 Play;
} TFT800Sound;

typedef struct
{
 uint32_ft8 StartAddress;
 uint32_ft8 Length;
 uint16_ft8 Frequency;
 uint8_ft8 Format;
 uint8_ft8 Loop;
 uint8_ft8 Volume;
 uint8_ft8 Play;
} TFT800Audio;

typedef struct
{

 uint16_ft8 Freq;
 uint8_ft8 Duty;
} TFT800PWM;

typedef struct
{
 uint8_ft8 GPIODIR;
 uint8_ft8 GPIO;
} TFT800GPIO;

typedef struct
{
```

```
 TFT800PWM *pPWMCfg;
 TFT800GPIO *pGPIOCfg;
 TFT800Audio *pAudioCfg;
 TFT800Sound *pSoundCfg;
 TFT800Touch *pTouchCfg;
 TFT800Display *pDisplayCfg;
 TFT800Interrupt *pInterruptCfg;
 TFT800TouchTransform *pTTransformCfg;
} TFT800Config;



typedef struct
{
 struct
 {




 TFT800Color Color;
 TFT800Alpha Alpha;

 uint8_ft8 Tag;

 TFT800Color ClearColor;
 TFT800Alpha ClearAlpha;
 uint8_ft8 ClearStencil;
 uint8_ft8 ClearTag;

 uint16_ft8 LineWidth;
 uint16_ft8 PointSize;

 uint16_ft8 ScissorLeft;
 uint16_ft8 ScissorTop;
 uint16_ft8 ScissorWidth;
 uint16_ft8 ScissorHeight;

 uint8_ft8 BmpHandle;
 uint8_ft8 Cell;

 struct
 {
 Color : 1;
 Alpha : 1;
 Tag : 1;
 ClearColor : 1;
 ClearAlpha : 1;
 ClearStencil : 1;
 ClearTag : 1;
 LineWidth : 1;
 PointSize : 1;
 ScissorPos : 1;
 ScissorSize : 1;
 BmpHandle : 1;
 Cell : 1;

 Unused : 32-13;

 } UpdateFlags;


 } Context;

 TFT800BmpConfig BmpHandleCfg[16];

 int8_ft8 CurrGrPrim;
} TFT800Graphics;

typedef struct
{
 TFT800IO IO;

 TFT800Sound Sound;

 TFT800Audio Audio;

 TFT800Graphics Graphics;

 struct
 { uint16_ft8 Width;
 uint16_ft8 Height;
 } Display;

 uint8_ft8 UpdateMode;

} TFT800Controller;
#line 6
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_objects.h"
typedef enum {taNone, taLeft, taCenter, taRight, taCenterX,
taCenterY, taRightX} TTextAlign;

typedef struct Screen TScreen;

typedef unsigned long TPointer;

typedef struct tagObjInfo {
 TPointer Obj;
 char Type;
 char Order;
 char Flags;

 char HitTag;
 int HitX;
 int HitY;
} TObjInfo;

typedef struct tagTouchStat {
 char Pressed;

 char Tag;
 int X;
 int Y;

 TObjInfo ActObjInfo;
} TTouchStat;

typedef void (*TDrawHandler)(TPointer aObj);

typedef void (*TEvtAction)();

typedef struct tagEvtSound {
 char SndAct;
 char Effect;
 char Pitch;
 char Volume;
} TEvtSound;

typedef const far struct tagCEvent {
 TEvtAction Action;
 TEvtSound Sound;
} TCEvent;

typedef struct tagEvent {
 TEvtAction Action;
 TEvtSound Sound;
} TEvent;
```

```
typedef const far struct tagCRect {
 int Left;
 int Top;
 int Width;
 int Height;
} TCRect;

typedef struct tagRect {
 int Left;
 int Top;
 int Width;
 int Height;
} TRect;

typedef struct tagBox {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char Gradient;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TBox;

typedef far const code struct tagCBox {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char Gradient;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCBox;

typedef struct tagBox_Round {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Corner_Radius;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TBox_Round;

typedef far const code struct tagCBox_Round {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Corner_Radius;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCBox_Round;

typedef struct tagLine {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int First_Point_X;
 int First_Point_Y;
 int Second_Point_X;
 int Second_Point_Y;
 char Pen_Width;
 unsigned int Pen_Color;
} TLine;

typedef struct tagEveGauge {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Radius;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Major;
 char Minor;
 unsigned int Value;
 unsigned int Range;
 char Flat;
 char NoBackground;
 char NoPointer;
```

```
 char TicksVisible;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveGauge;

typedef struct tagEveKeys {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char *Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Flat;
 char AutoSize;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveKeys;

typedef struct tagEveProgressBar {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Background_Color;
 unsigned int Color;
 unsigned int Value;
 unsigned int Range;
 char Flat;
} TEveProgressBar;

typedef struct tagEveToggle {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Background_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char *StateOFF_Caption;
 char *StateON_Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 unsigned int State;
 char Flat;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveToggle;

typedef struct tagEveButton {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char *Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Flat;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveButton;

typedef struct tagEveText {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char *Caption;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveText;

typedef far const code struct tagCEveText {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
```

```
 int Top;
 int Width;
 int Height;
 far const code char *Caption;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCEveText;

typedef far const code struct tagCEveNumber {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Text_Length;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 signed long Value;
 unsigned char Signed;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCEveNumber;

struct Screen {
 unsigned int Color;
 unsigned int Width;
 unsigned int Height;
 unsigned short ObjectsCount;
 unsigned short BoxesCount;
 TBox *far const code *Boxes;
 unsigned short CBoxesCount;
 TCBox *far const code *CBoxes;
 unsigned short Boxes_RoundCount;
 TBox_Round *far const code *Boxes_Round;
 unsigned short CBoxes_RoundCount;
 TCBox_Round *far const code *CBoxes_Round;
 unsigned short LinesCount;
 TLine *far const code *Lines;
 unsigned short EveGaugesCount;
 TEveGauge *far const code *EveGauges;
 unsigned short EveKeysCount;
 TEveKeys *far const code *EveKeys;
 unsigned short EveProgressBarsCount;
 TEveProgressBar *far const code *EveProgressBars;
 unsigned short EveTogglesCount;
 TEveToggle *far const code *EveToggles;
 unsigned short EveButtonsCount;
 TEveButton *far const code *EveButtons;
 unsigned short EveTextsCount;
 TEveText *far const code *EveTexts;
 unsigned short CEveTextsCount;
 TCEveText *far const code *CEveTexts;
 unsigned short CEveNumbersCount;
 TCEveNumber *far const code *CEveNumbers;
 unsigned long DynResStart;
 unsigned short Active;
 unsigned short SniffObjectEvents;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnTagChange;
 TEvent *OnPress;
};

extern const VTFT_OT_BOX;
extern const VTFT_OT_CBOX;
extern const VTFT_OT_BOXROUND;
extern const VTFT_OT_CBOXROUND;
extern const VTFT_OT_LINE;
extern const VTFT_OT_EVEGAUGE;
extern const VTFT_OT_EVEKEYS;
extern const VTFT_OT_EVEPROGRESSBAR;
extern const VTFT_OT_EVETOGGLE;
extern const VTFT_OT_EVEBUTTON;
extern const VTFT_OT_EVETEXT;
extern const VTFT_OT_CEVETEXT;
extern const VTFT_OT_CEVENUMBER;

extern const VTFT_EVT_UP;
extern const VTFT_EVT_DOWN;
extern const VTFT_EVT_CLICK;
extern const VTFT_EVT_PRESS;

extern const VTFT_SNDACT_NONE;
extern const VTFT_SNDACT_PLAY;
extern const VTFT_SNDACT_STOP;

extern const VTFT_LOAD_RES_NONE;
extern const VTFT_LOAD_RES_STATIC;
extern const VTFT_LOAD_RES_DYNAMIC;
extern const VTFT_LOAD_RES_ALL;

extern const VTFT_DISPLAY_EFF_NONE;
extern const VTFT_DISPLAY_EFF_LIGHTS_FADE;
extern const VTFT_DISPLAY_EFF_LIGHTS_OFF;

extern const VTFT_INT_REPAINT_ON_DOWN;
extern const VTFT_INT_REPAINT_ON_UP;
extern const VTFT_INT_BRING_TO_FRONT;
extern const VTFT_INT_INTRINSIC_CLICK_EFF;

extern const TPointer DrawHandlerTable[44];
```

```
extern const TFT800PWM VTFT_FT800_CONFIG_PWM;
extern const TFT800GPIO VTFT_FT800_CONFIG_GPIO;
extern const TFT800Sound VTFT_FT800_CONFIG_SOUND;
extern const TFT800Audio VTFT_FT800_CONFIG_AUDIO;
extern const TFT800Display VTFT_FT800_CONFIG_DISPLAY;
extern const TFT800Interrupt
VTFT_FT800_CONFIG_INTERRUPT;
extern const TFT800Touch VTFT_FT800_CONFIG_TOUCH;
extern const TFT800TouchTransform
VTFT_FT800_CONFIG_TOUCHTRANSFORM;

extern TTouchStat TouchS;

extern TScreen SplashScreen;

extern TCEveText EveTextSplashScreenName;
extern TEveButton EveButtonSplashInit;
extern TEvent EveButtonSplashInit_OnClick;
extern TEveText EveTextSplashScreenStatus;
extern TEveProgressBar EveProgressBar1;

extern TEveProgressBar *far const code
SplashScreen_EveProgressBars[1];
extern TEveButton *far const code SplashScreen_EveButtons[1];
extern TEveText *far const code SplashScreen_EveTexts[1];
extern TCEveText *far const code SplashScreen_CEveTexts[1];

extern TScreen SamplingScreen;

extern TCBox_Round BoxRoundsmpScn1;
extern TCEveText EveTextSampScnLabel1;
extern TCEveText EveTextsampscnlabel2;
extern TCEveText EveTextsampscnlabel4;
extern TCEveText EveTextsampscnlabel3;
extern TCEveText EveTextsampscnlabel5;
extern TCEveText EveTextsampscnlabel6;
extern TCBox_Round BoxRoundSmpscn2;
extern TEveText EveTextEncoderSamp;
extern TEveText EveTextLaserSmp;
extern TEveText EveTextPitchSmp;
extern TEveText EveTextGPSSmp;
extern TEveText EveTextSDSmp;
extern TEveText EveTextRollSmp;
extern TEveText EveTextAccx;
extern TEveText EveTextAccy;
extern TEveText EveTextAccz;
extern TBox_Round BoxRound7;
extern TEveButton EveButtonsmpJogRear;
extern TEvent EveButtonsmpJogRear_OnPress;
extern TEveButton EveButtonsmpJogFront;
extern TEvent EveButtonsmpJogFront_OnPress;
extern TEveButton EveButtonsmpBack;
extern TEvent EveButtonsmpBack_OnClick;
extern TBox_Round BoxRound8;
extern TEveButton EveButtonSmpStartSampling;
extern TEvent EveButtonSmpStartSampling_OnClick;
extern TEveButton EveButtonSmpStopSampling;
extern TEvent EveButtonSmpStopSampling_OnClick;
extern TEveButton EveButtonSmpConfigureMotor;
extern TEvent EveButtonSmpConfigureMotor_OnClick;
extern TEveButton EveButtonSmpGotoProf;
extern TEvent EveButtonSmpGotoProf_OnClick;
extern TEveButton EveButtonsmpResetInc;
extern TEvent EveButtonsmpResetInc_OnClick;

extern TBox_Round *far const code
SamplingScreen_Boxes_Round[2];
extern TCBox_Round *far const code
SamplingScreen_CBoxes_Round[2];
extern TEveButton *far const code
SamplingScreen_EveButtons[8];
extern TEveText *far const code SamplingScreen_EveTexts[9];
extern TCEveText *far const code SamplingScreen_CEveTexts[6];

extern TScreen ProfilingScreen;

extern TCBox Box1;
extern TCBox Box2;
extern TCEveText EveTextProflabel1;
extern TCEveText EveTextProflabel2;
extern TCEveNumber EveNumber1;
extern TCEveNumber EveNumber2;
extern TCEveNumber EveNumber3;
extern TCEveNumber EveNumber4;
extern TCEveNumber EveNumber5;
extern TCEveNumber EveNumber6;
extern TCBox_Round BoxRound1;
extern TEveGauge EveGaugeProfSpeed;
extern TEveGauge EveGaugeProfCrossSlp;
extern TCEveText EveTextProflabel5;
extern TCEveText EveTextProflabel4;
extern TEveToggle EveToggleProfMode;
extern TEvent EveToggleProfMode_OnClick;
extern TCEveText EveTextProflabel3;
extern TEveButton EveButtonProfStartProfiling;
extern TEvent EveButtonProfStartProfiling_OnClick;
extern TEveButton EveButtonProfStopProfiling;
extern TEvent EveButtonProfStopProfiling_OnClick;
extern TLine Line1;
extern TLine Line2;
extern TLine Line3;
extern TLine Line4;
extern TLine Line5;
extern TLine Line6;
extern TLine Line7;
extern TLine Line8;
extern TLine Line9;
extern TLine Line10;
extern TLine Line11;
extern TLine Line12;
extern TLine Line13;
extern TLine Line14;
extern TLine Line15;
extern TLine Line16;
extern TLine Line17;
extern TLine Line18;
extern TLine Line19;
extern TLine Line20;
extern TLine Line21;
extern TLine Line22;
extern TLine Line23;
extern TLine Line24;
extern TLine Line25;
extern TLine Line26;
extern TLine Line27;
extern TLine Line28;
extern TLine Line29;
extern TLine Line30;
extern TLine Line31;
extern TLine Line32;
extern TLine Line33;
extern TLine Line34;
extern TLine Line35;
extern TLine Line36;
extern TLine Line37;
extern TLine Line38;
extern TLine Line39;
```

```
extern TLine Line40;
extern TLine Line41;
extern TLine Line42;
extern TLine Line43;
extern TLine Line44;
extern TLine Line45;
extern TLine Line46;
extern TLine Line47;
extern TLine Line48;
extern TLine Line49;
extern TLine Line50;
extern TEveText EveText1;
extern TEveButton EveButtonProfCreateFile;
extern TEvent EveButtonProfCreateFile_OnClick;
extern TCEveText EveText10;
extern TEveText EveTextProfFileName;
extern TCEveText EveText12;
extern TBox BoxProfWaitingForCam;
extern TEveText EveTextProfWaitingForCam;

extern TBox *far const code ProfilingScreen_Boxes[1];
extern TCBox *far const code ProfilingScreen_CBoxes[2];
extern TCBox_Round *far const code
ProfilingScreen_CBoxes_Round[1];
extern TLine *far const code ProfilingScreen_Lines[50];
extern TEveGauge *far const code ProfilingScreen_EveGauges[2];
extern TEveToggle *far const code
ProfilingScreen_EveToggles[1];
extern TEveButton *far const code
ProfilingScreen_EveButtons[3];
extern TEveText *far const code ProfilingScreen_EveTexts[3];
extern TCEveText *far const code ProfilingScreen_CEveTexts[7];
extern TCEveNumber *far const code
ProfilingScreen_CEveNumbers[6];

extern TScreen SettingScreen;
extern TEvent SettingScreen_OnTagChange;

extern TEveButton EveButtonFNfilename;
extern TCBox_Round BoxRound2;
extern TEveKeys EveKeys1;
extern TEveKeys EveKeys2;
extern TEveKeys EveKeys3;
extern TEveKeys EveKeys4;
extern TCEveText EveText2;
extern TCBox_Round BoxRound3;
extern TEveButton EveButtonFNBackspace;
extern TEvent EveButtonFNBackspace_OnClick;
extern TEveButton EveButtonFNClear;
extern TEvent EveButtonFNClear_OnClick;
extern TEveButton EveButtonFNConfirm;
extern TEvent EveButtonFNConfirm_OnClick;
extern TEveButton EveButtonFNGoBack;
extern TEvent EveButtonFNGoBack_OnClick;
extern TEveButton EveButtonFNShift;
extern TEvent EveButtonFNShift_OnClick;

extern TCBox_Round *far const code
SettingScreen_CBoxes_Round[2];
extern TEveKeys *far const code SettingScreen_EveKeys[4];
extern TEveButton *far const code SettingScreen_EveButtons[6];
extern TCEveText *far const code SettingScreen_CEveTexts[1];

extern TScreen SummaryScreen;

extern TCBox_Round BoxRound4;
extern TCEveText EveText3;
extern TCEveText EveText4;
extern TCEveText EveText5;
extern TCEveText EveText6;
extern TCEveText EveText7;
extern TCBox_Round BoxRound5;
extern TEveButton EveButtonSUNewRun;
extern TEvent EveButtonSUNewRun_OnClick;
extern TEveButton EveButtonSUHelp;
extern TEvent EveButtonSUHelp_OnClick;
extern TEveButton EveButtonSUAbout;
extern TEvent EveButtonSUAbout_OnClick;
extern TEveText EveTextSUtime;
extern TEveText EveTextSUdistance;
extern TEveText EveTextSUSpeed;
extern TEveText EveTextSUFilename;
extern TEveText EveTextSUAverageSpeed;
extern TEveText EveText8;
extern TBox_Round BoxRound6;
extern TEveText EveText9;
extern TEveText EveText11;
extern TEveText EveText13;
extern TEveText EveText14;
extern TEveText EveText15;
extern TEveText EveText16;
extern TEveText EveText17;
extern TEveButton EveButtonSUBack;
extern TEvent EveButtonSUBack_OnClick;

extern TBox_Round *far const code
SummaryScreen_Boxes_Round[1];
extern TCBox_Round *far const code
SummaryScreen_CBoxes_Round[2];
extern TEveButton *far const code
SummaryScreen_EveButtons[4];
extern TEveText *far const code SummaryScreen_EveTexts[13];
extern TCEveText *far const code
SummaryScreen_CEveTexts[5];

extern TScreen *CurrentScreen;

void EveButtonFNBackspaceOnClick();
void EveButtonFNClearOnClick();
void EveButtonFNConfirmOnClick();
void EveButtonFNGoBackOnClick();
void EveButtonFNShiftOnClick();
void EveButtonProfCreateFileOnClick();
void EveButtonProfStartProfilingOnClick();
void EveButtonProfStopProfilingOnClick();
void EveButtonsmpBackOnClick();
void EveButtonSmpConfigureMotorOnClick();
void EveButtonSmpGotoProfOnClick();
void EveButtonsmpJogFrontOnPress();
void EveButtonsmpJogRearOnPress();
void EveButtonsmpResetIncOnClick();
void EveButtonSmpStartSamplingOnClick();
void EveButtonSmpStopSamplingOnClick();
void EveButtonSplashInitOnClick();
void EveButtonSUAboutOnClick();
void EveButtonSUBackOnClick();
void EveButtonSUHelpOnClick();
void EveButtonSUNewRunOnClick();
void EveToggleProfModeOnClick();
void SettingScreenOnTagChange();

extern const code far char EveTextSplashScreenName_Caption[];
extern char EveButtonSplashInit_Caption[];
extern char EveTextSplashScreenStatus_Caption[];
extern char EveProgressBar1_Caption[];
```

```
extern const code far char BoxRoundsmpScn1_Caption[];
extern const code far char EveTextSampScnLabel1_Caption[];
extern const code far char EveTextsampscnlabel2_Caption[];
extern const code far char EveTextsampscnlabel4_Caption[];
extern const code far char EveTextsampscnlabel3_Caption[];
extern const code far char EveTextsampscnlabel5_Caption[];
extern const code far char EveTextsampscnlabel6_Caption[];
extern const code far char BoxRoundSmpscn2_Caption[];
extern char EveTextEncoderSamp_Caption[];
extern char EveTextLaserSmp_Caption[];
extern char EveTextPitchSmp_Caption[];
extern char EveTextGPSSmp_Caption[];
extern char EveTextSDSmp_Caption[];
extern char EveTextRollSmp_Caption[];
extern char EveTextAccx_Caption[];
extern char EveTextAccy_Caption[];
extern char EveTextAccz_Caption[];
extern char BoxRound7_Caption[];
extern char EveButtonsmpJogRear_Caption[];
extern char EveButtonsmpJogFront_Caption[];
extern char EveButtonsmpBack_Caption[];
extern char BoxRound8_Caption[];
extern char EveButtonSmpStartSampling_Caption[];
extern char EveButtonSmpStopSampling_Caption[];
extern char EveButtonSmpConfigureMotor_Caption[];
extern char EveButtonSmpGotoProf_Caption[];
extern char EveButtonsmpResetInc_Caption[];
extern const code far char Box1_Caption[];
extern const code far char Box2_Caption[];
extern const code far char EveTextProflabel1_Caption[];
extern const code far char EveTextProflabel2_Caption[];
extern const code far char BoxRound1_Caption[];
extern char EveGaugeProfSpeed_Caption[];
extern char EveGaugeProfCrossSlp_Caption[];
extern const code far char EveTextProflabel5_Caption[];
extern const code far char EveTextProflabel4_Caption[];
extern char EveToggleProfMode_StateOFF_Caption[];
extern char EveToggleProfMode_StateON_Caption[];
extern const code far char EveTextProflabel3_Caption[];
extern char EveButtonProfStartProfiling_Caption[];
extern char EveButtonProfStopProfiling_Caption[];
extern char Line1_Caption[];
extern char Line2_Caption[];
extern char Line3_Caption[];
extern char Line4_Caption[];
extern char Line5_Caption[];
extern char Line6_Caption[];
extern char Line7_Caption[];
extern char Line8_Caption[];
extern char Line9_Caption[];
extern char Line10_Caption[];
extern char Line11_Caption[];
extern char Line12_Caption[];
extern char Line13_Caption[];
extern char Line14_Caption[];
extern char Line15_Caption[];
extern char Line16_Caption[];
extern char Line17_Caption[];
extern char Line18_Caption[];
extern char Line19_Caption[];
extern char Line20_Caption[];
extern char Line21_Caption[];
extern char Line22_Caption[];
extern char Line23_Caption[];
extern char Line24_Caption[];
extern char Line25_Caption[];
extern char Line26_Caption[];
extern char Line27_Caption[];
extern char Line28_Caption[];
extern char Line29_Caption[];
extern char Line30_Caption[];
extern char Line31_Caption[];
extern char Line32_Caption[];
extern char Line33_Caption[];
extern char Line34_Caption[];
extern char Line35_Caption[];
extern char Line36_Caption[];
extern char Line37_Caption[];
extern char Line38_Caption[];
extern char Line39_Caption[];
extern char Line40_Caption[];
extern char Line41_Caption[];
extern char Line42_Caption[];
extern char Line43_Caption[];
extern char Line44_Caption[];
extern char Line45_Caption[];
extern char Line46_Caption[];
extern char Line47_Caption[];
extern char Line48_Caption[];
extern char Line49_Caption[];
extern char Line50_Caption[];
extern char EveText1_Caption[];
extern char EveButtonProfCreateFile_Caption[];
extern const code far char EveText10_Caption[];
extern char EveTextProfFileName_Caption[];
extern const code far char EveText12_Caption[];
extern char BoxProfWaitingForCam_Caption[];
extern char EveTextProfWaitingForCam_Caption[];
extern char EveButtonFNfilename_Caption[];
extern const code far char BoxRound2_Caption[];
extern char EveKeys1_Caption[];
extern char EveKeys2_Caption[];
extern char EveKeys3_Caption[];
extern char EveKeys4_Caption[];
extern const code far char EveText2_Caption[];
extern const code far char BoxRound3_Caption[];
extern char EveButtonFNBackspace_Caption[];
extern char EveButtonFNClear_Caption[];
extern char EveButtonFNConfirm_Caption[];
extern char EveButtonFNGoBack_Caption[];
extern char EveButtonFNShift_Caption[];
extern const code far char BoxRound4_Caption[];
extern const code far char EveText3_Caption[];
extern const code far char EveText4_Caption[];
extern const code far char EveText5_Caption[];
extern const code far char EveText6_Caption[];
extern const code far char EveText7_Caption[];
extern const code far char BoxRound5_Caption[];
extern char EveButtonSUNewRun_Caption[];
extern char EveButtonSUHelp_Caption[];
extern char EveButtonSUAbout_Caption[];
extern char EveTextSUtime_Caption[];
extern char EveTextSUdistance_Caption[];
extern char EveTextSUSpeed_Caption[];
extern char EveTextSUFilename_Caption[];
extern char EveTextSUAverageSpeed_Caption[];
extern char EveText8_Caption[];
extern char BoxRound6_Caption[];
extern char EveText9_Caption[];
extern char EveText11_Caption[];
extern char EveText13_Caption[];
extern char EveText14_Caption[];
extern char EveText15_Caption[];
extern char EveText16_Caption[];
extern char EveText17_Caption[];
extern char EveButtonSUBack_Caption[];

extern TEvent EveButtonSplashInit_OnUpOnClick;
```

```
extern TEvent EveButtonsmpJogRear_OnUpOnPress;
extern TEvent EveButtonsmpJogFront_OnUpOnPress;
extern TEvent EveButtonsmpBack_OnUpOnClick;
extern TEvent EveButtonSmpStartSampling_OnUpOnClick;
extern TEvent EveButtonSmpStopSampling_OnUpOnClick;
extern TEvent EveButtonSmpConfigureMotor_OnUpOnClick;
extern TEvent EveButtonSmpGotoProf_OnUpOnClick;
extern TEvent EveButtonsmpResetInc_OnUpOnClick;
extern TEvent EveToggleProfMode_OnUpOnClick;
extern TEvent EveButtonProfStartProfiling_OnUpOnClick;
extern TEvent EveButtonProfStopProfiling_OnUpOnClick;
extern TEvent EveButtonProfCreateFile_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNBackspace_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNClear_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNConfirm_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNGoBack_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNShift_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonSUNewRun_OnUpOnClick;
extern TEvent EveButtonSUHelp_OnUpOnClick;
extern TEvent EveButtonSUAbout_OnUpOnClick;
extern TEvent EveButtonSUBack_OnUpOnClick;


void DrawScreenO(TScreen *aScreen, char aOptions);
void DrawScreen(TScreen *aScreen);
void DrawBox(TBox *ABox);
void DrawCBox(TCBox *ACBox);
void DrawBoxRound(TBox_Round *ABoxRound);
void DrawCBoxRound(TCBox_Round *ACBoxRound);
void DrawLine(TLine *ALine);
void DrawEveGauge(TEveGauge *AEveGauge);
void DrawEveKeys(TEveKeys *AEveKeys);
void DrawEveProgressBar(TEveProgressBar *AEveProgressBar);
void DrawEveToggle(TEveToggle *AEveToggle);
void DrawEveButton(TEveButton *AEveButton);
void DrawEveText(TEveText *AEveText);
void DrawCEveText(TCEveText *ACEveText);
void DrawCEveNumber(TCEveNumber *ACEveNumber);
void ProcessVTFTStack();
void InitVTFTStack();
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_resources.h"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for dspic/fandh.h"


float wheelRadius=0.2791;
unsigned int cycle=2400;

void Initplot(TLine *far const code lines[],unsigned int
numberOfLines,unsigned int Xbase,unsigned int Ybase,unsigned
int lineResolution,int *pf);
void addData(int Newdata,int *pf,int Ybase,TLine *far const code
lines[],unsigned int numberOfLines);

float calculateLaz(unsigned char *buff);

float CalculateAngle(unsigned char *angleBuff);
unsigned long calculateRQEI(unsigned char *buff);
float CalculateSpeed(unsigned char *tBuff, unsigned char
*encBuff);


void initCommunication();
void mapPins();



volatile unsigned int laserplotflag=0;
volatile unsigned int Incplotflag=0;

volatile unsigned int smpflag=0;
volatile unsigned int loggingflag=0;
volatile unsigned int profilingMode=0;



volatile float laserMeasurement=0;
volatile float speed=0;
volatile unsigned speedCounter=0;
volatile float averageSpeed=0;
volatile unsigned char smpbuff[30];
volatile unsigned int smpbuffReady=0;
volatile unsigned int smpbuffIterator=0;

volatile float Pitch=0;
volatile float roll=0;

volatile unsigned long oldPos=0;
volatile unsigned long oldT=0;
volatile unsigned long newPos=0;
volatile unsigned long newT=0;


volatile unsigned char GPSpackage[90];
volatile unsigned int GPSSize=0;


unsigned char foldername[9];
unsigned char oldname[11];

volatile unsigned char GPSdataReceived[30];
volatile unsigned int GPSdataReceivedPtr=0;
#line 6
"C:/Users/ipm4/Desktop/latest_code_01062014/0121/NewDesignF
T800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/NewDesign_events_code.c"
void mapPins(){
 TRISB.B12=1;
 TRISB.B11=0;
 TRISB.B10=0;

 TRISB.B4=0;
 TRISA.B4=1;

 PPS_Mapping(44,_INPUT,_SDI2IN);
 PPS_Mapping(43,_OUTPUT,_SDO2);
 PPS_Mapping(42,_OUTPUT,_SCK2OUT);

 PPS_Mapping(36,_OUTPUT,_U1TX);
 PPS_Mapping(20,_INPUT,_U1RX);
```

```
}
void initCommunication(){
 unsigned int i=0;


UART1_Init_Advanced(921600,_UART_8BIT_NOPARITY,_UA
RT_ONE_STOPBIT,_UART_HI_SPEED);
 Delay_ms(150);
 IFS0.B11=0;
 U1STA.OERR=0;
 IPC2.B14=1;
 IPC2.B13=0;
 IPC2.B12=0;

 IFS0.B11=0;
 IEC0.B11=1;

 Delay_ms(100);
 SPI2_Init_Advanced(_SPI_MASTER,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_4,
 _SPI_PRESCALE_PRI_16,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_MIDDLE,
 _SPI_CLK_IDLE_LOW,
 _SPI_ACTIVE_2_IDLE);
 SPI_Set_Active(SPI1_Read, SPI1_Write);
 for(i=0;i<30;i++){
 smpbuff[i]=0x00;
 }
}
void SPI2Interrupt() iv IVT_ADDR_SPI2INTERRUPT{
 unsigned char temp;
 IFS2.B1=0;
 temp=SPI2BUF;

 GPSdataReceived[GPSdataReceivedPtr++]=temp;
 if(temp==0x00){

 GPSdataReceivedPtr=0;

 return;
 }
 else if(GPSdataReceivedPtr==35){GPSdataReceivedPtr=0;}


}
void smpInterrupt() iv IVT_ADDR_U1RXINTERRUPT{
 unsigned char temp1=0;
 static unsigned int count;
 IFS0.B11=0;
 temp1=U1RXREG;

 if(temp1==0xAA){
 count++;
 if(count==8){
 count=0;
 laserplotflag=0;
 Incplotflag=0;
 return;
 }
 }
 else{
 count=0;
 }



 if(smpflag){
 smpbuff[smpbuffIterator++]=temp1;
 if(smpbuffIterator==22){
 smpbuffIterator=0;
 smpbuffReady=1;
 }
 }
 if(loggingflag){
 smpbuff[smpbuffIterator++]=temp1;
 if(smpbuffIterator==19){
 smpbuffIterator=0;
 pitch=CalculateAngle(&smpbuff[4]);
 roll= CalculateAngle(&smpbuff[8]);
 speed=CalculateSpeed(&smpbuff[12],&smpbuff[15]);
 smpbuffReady=1;
 }

 }


}
unsigned long calculateRQEI(unsigned char *buff){
 unsigned long result;
 unsigned long d;
 unsigned long c;
 unsigned int b;
 d=*buff;
 d=d<<24;
 c=*(buff+1);
 c=c<<16;
 b=*(buff+2);
 b=b<<8;
 d=d|c|b|(*(buff+3));
 return d;

}
float calculateLaz(unsigned char *buff){
 return
(((*buff)<<12)|((*(buff+1))<<8)|((*(buff+2))<<4)|((*(buff+3))))/1
6463f*19.5f+5f;

}

float CalculateAngle(unsigned char *angleBuff){
 static unsigned long a;
 static unsigned long b;
 static unsigned int c;
 a=angleBuff[0];
 b=angleBuff[1];
 c=angleBuff[2];

 if(a!=0xff){
 return (((a<<24)|(b<<16)|(c<<8)|(angleBuff[3]))/1000.0);
 }
 else{return (((a<<24)|(b<<16)|(c<<8)|(angleBuff[3])-
0xffffffff))/1000.0;}
}
float CalculateAcc(unsigned char *accBuff){
 unsigned int temp;
 temp=accBuff[0];
 temp=temp<<8;
 temp=temp|accBuff[1];
 return (temp/1023.0*3.3-1.65)/0.3;
}

unsigned long calculateTime(unsigned char *tBuff){
 return (tBuff[0]<<16)|(tBuff[1]<<8)|(tBuff[2]);
}

float CalculateSpeed(unsigned char *tBuff, unsigned char
*encBuff){
 float speedtemp=0;
```

```
 newPos=calculateRQEI(encBuff);
 newT=calculateTime(tBuff);
 speedtemp=(newPos-oldPos)*wheelRadius*6.28318/cycle;
 speedtemp=speedtemp/((newT-oldT)/1000.0);
 oldT=newT;
 oldPos=newPos;
 speedCounter++;
 averageSpeed+=speedtemp;
 return speedtemp;
}
void Initplot(TLine *far const code lines[],unsigned int
numberOfLines,unsigned int Xbase,unsigned int Ybase,unsigned
int lineResolution,int *pf){
 int i=0;

 lines[0]->First_Point_X=Xbase;
 lines[0]->Second_Point_X=Xbase+lineResolution;
 lines[0]->First_Point_Y=Ybase;
 lines[0]->Second_Point_Y=Ybase;

 lines[0]->visible=1;
 for(i=1;i<numberOfLines;i++){
 lines[i]->First_Point_X=lines[i-1]->Second_Point_X;
 lines[i]->First_Point_Y=lines[i-1]->Second_Point_Y;
 lines[i]->Second_Point_X=lines[i]-
>First_Point_X+lineResolution;
 lines[i]->Second_Point_Y=lines[i]->First_Point_Y;
 lines[i]->visible=1;
 }
 *pf=1;
 DrawScreen(&ProfilingScreen);
}
void addData(int Newdata,int *pf,int Ybase,TLine *far const code
lines[],unsigned int numberOfLines){
 int i=0;
 if(*pf){
 for(i=0;i<numberOfLines-1;i++){
 lines[i]->First_Point_Y=lines[i+1]->First_Point_Y;
 lines[i]->Second_Point_Y=lines[i+1]->Second_Point_Y;
 }
 lines[numberOfLines-1]->First_Point_Y=lines[numberOfLines-
2]->Second_Point_Y;
 lines[numberOfLines-1]->Second_Point_Y=Ybase+newData;
 }
}
void sendCommandToCamChip(unsigned char* name,unsigned
char size){

 unsigned int i=0;


 SPI2BUF=(0x53);


 Delay_us(100);
 for(i=0;i<size;i++){
 SPI2BUF=(name[i]);
 Delay_us(100);
 }
 SPI2BUF=(0xE4);
 Delay_us(100);
 SPI2BUF=(0x43);
 Delay_us(100);

}
void sendNameToSamChip(unsigned char* name,unsigned char
size){
 unsigned int i=0;
 UART1_Write(0xE3);
 for(i=0;i<size;i++){
 UART1_Write(name[i]);
 }
 UART1_Write(0xB5);
}

void checkFileName(){
 if(strcmp(EveButtonFNfilename_Caption,oldname)==0){
 EveButtonFNConfirm.Active=0;
 EveButtonFNConfirm.Color=0xAD55;
 }
 else{
 EveButtonFNConfirm.Active=1;
 EveButtonFNConfirm.Color=0x3666;
 }
}



void EveButtonSmpStartSamplingOnClick() {
 unsigned int i=0;
 unsigned int b=0;
 unsigned char temp[3];
 unsigned char te;

 smpbuffIterator=0;
 smpbuffReady=0;
 EveButtonSmpConfigureMotor.color=0xAD55;
 EveButtonSmpConfigureMotor.active=0;


 EveButtonSmpStartSampling.visible=0;
 EveButtonSmpStartSampling.Active=0;
 EveButtonSmpStopSampling.visible=1;
 EveButtonSmpStopSampling.Active=1;
 DrawScreen(&SamplingScreen);

 smpflag=1;
 SPI2STAT.B4=0;
 SPI2STAT.B3=0;
 SPI2STAT.B2=1;
 IFS2.B1=0;
 IPC8.B6=1;
 IPC8.B5=0;
 IPC8.B4=0;
 IEC2.B1=1;

 SPI2BUF=0x31;
 Delay_ms(1);
 GPSdataReceivedPtr=0;
 UART1_Write(0x31);
 Delay_ms(100);

 while(smpflag){
 for(b=0;b<35;b++){
 SPI2BUF=0x31;
 Delay_us(100);
 }
 ProcessVTFTStack();
 Delay_ms(10);
 if(smpbuffReady){
 smpbuffReady=0;
```

```
 sprintf(EveTextLaserSmp.Caption, "%.4f",
calculateLaz(&smpbuff));
 sprintf(EveTextEncoderSamp.Caption,
"%+ld",calculateRQEI(&smpbuff[4]));
 sprintf(EveTextPitchSmp.Caption,
"%+.2f",CalculateAngle(&smpbuff[8]));
 sprintf(EveTextRollSmp.Caption,
"%+.2f",CalculateAngle(&smpbuff[12]));



 EveTextGPSSmp.Caption=GPSdataReceived;
 sprintf(EveTextAccx_Caption, "%+.2f",
CalculateAcc(&smpbuff[16]));
 sprintf(EveTextAccy_Caption, "%+.2f",
CalculateAcc(&smpbuff[18]));
 sprintf(EveTextAccz_Caption, "%+.2f",
CalculateAcc(&smpbuff[20]));

 UART1_Write(0x31);
 DrawScreen(&SamplingScreen);
 }


 }
 EveButtonSmpStartSampling.visible=1;
 EveButtonSmpStartSampling.Active=1;
 EveButtonSmpStopSampling.visible=0;
 EveButtonSmpStopSampling.Active=0;

 SPI2BUF=0x22;


 EveButtonSmpConfigureMotor.color=0xA865;
 EveButtonSmpConfigureMotor.active=1;
 DrawScreen(&SamplingScreen);
 Delay_ms(500);
 IEC2.B1=1;
 GPSdataReceivedPtr=0;
 smpbuffIterator=0;
 smpbuffReady=0;
}

void EveButtonSmpStopSamplingOnClick() {
 smpbuffIterator=0;
 smpflag=0;

}

void EveButtonSplashInitOnClick() {
 int i=0;
 unsigned int pwm_period1;

 EveButtonSplashInit.visible=0;
 EveProgressBar1.visible=1;
 EveTextSplashScreenStatus.visible=1;

 mapPins();
 TRISB.B14=0;
 LATB.B14=0;
 initCommunication();



for(i=0;i<100;i++){EveProgressBar1.value=i;DrawScreen(&Splas
hScreen);Delay_ms(200);}
 DrawScreen(&SamplingScreen);
}

void EveButtonSmpGotoProfOnClick() {
 EveButtonProfStartProfiling.Active=0;
 EveButtonProfStartProfiling.Color=0xAD55;



 DrawScreen(&ProfilingScreen);
}

void EveButtonProfStartProfilingOnClick() {
 int crossslp;
 strcpy(oldname,EveTextProfFileName_Caption);
 Delay_ms(200);
 smpflag=0;
 smpbuffIterator=0;
 smpbuffReady=0;


 EveButtonProfStartProfiling.visible=0;
 EveButtonProfStartProfiling.Active=0;
 EveButtonProfStopProfiling.visible=1;
 EveButtonProfStopProfiling.Active=1;
 EveToggleProfMode.Active=0;



sendCommandToCamChip(EveButtonFNfilename.Caption,strlen(
EveButtonFNfilename.Caption)+1);

 DrawScreen(&ProfilingScreen);


sendNameToSamChip(EveButtonFNfilename_Caption,strlen(Eve
ButtonFNfilename_Caption)+1);



 Initplot(ProfilingScreen_Lines,25,34,80,8,&laserplotflag);

 Initplot(ProfilingScreen_Lines+25,25,34,172,8,&incplotflag);
 loggingflag=1;
 if(profilingMode){UART1_Write(0x1C);}
 Delay_ms(1);
 UART1_Write(0xA2);
 Delay_ms(500);


 while(laserplotflag){
 if(smpbuffReady){
 smpbuffReady=0;
 laserMeasurement=CalculateLaz(&smpbuff);

 if(laserMeasurement>7){
 laserMeasurement=7;
 }
 if(Pitch>50||Pitch<-50){pitch=0;}
 if(Roll>50||Roll<-50){Roll=0;}
 if(Pitch>3){Pitch=3;}
 else if(Pitch<-3){Pitch=-3;}
 if(Roll>30){Roll=30;}
 else if(Roll<-30){Roll=-30;}

 addData((laserMeasurement-
5)*50,&laserplotflag,24,ProfilingScreen_Lines,25);
```

```
addData(Pitch*8.3,&incplotflag,177,ProfilingScreen_Lines+25,25)
;

 crossslp=Roll+50;
 EveGaugeProfCrossSlp.value=crossslp;
 EveGaugeProfSpeed.value=speed*50;
 if(speed>1.99){speed=2;}
 if(speed<0.3||speed>1.7){EveGaugeProfSpeed.Color= 0xC800;}
 else{EveGaugeProfSpeed.Color= 0x03DA; }
 DrawScreen(&ProfilingScreen);
 }
 ProcessVTFTStack();
 }

 BoxProfWaitingForCam.visible=1;
 EveTextProfWaitingForCam.visible=1;
 EveButtonProfStartProfiling.visible=1;
 EveButtonProfStartProfiling.Active=1;
 EveButtonProfStopProfiling.visible=0;
 EveButtonProfStopProfiling.Active=0;
 EveToggleProfMode.Active=1;
 SPI2BUF=0x62;
 UART1_Write(0xB7);
 DrawScreen(&ProfilingScreen);
 Delay_ms(4000);

 sprintf(EveTextSUdistance.Caption, "%.2f",
calculateRQEI(&smpbuff[15])*wheelRadius*6.28318/cycle);
 sprintf(EveTextSUtime.Caption,
"%.2f",calculateTime(&smpbuff[12])/1000.0);
 sprintf(EveTextSUSpeed.Caption, "%.2f",speed);
 sprintf(EveTextSUAverageSpeed.Caption,
"%.2f",averageSpeed/speedCounter);
 EveTextSUFilename.Caption= EveButtonFNfilename.Caption;
 oldPos=0;
 oldT=0;



 DrawScreen(&SummaryScreen);

 loggingflag=0;
}

void EveButtonProfStopProfilingOnClick() {

 laserplotflag=0;
 Incplotflag=0;

}

void EveToggleProfModeOnClick() {
 profilingMode=EveToggleProfMode.state;

}


void EveButtonProfCreateFileOnClick() {
 EveButtonFNConfirm.Active=0;

 EveButtonFNConfirm.Color=0xAD55;
 DrawScreen(&SettingScreen);
}

void SettingScreenOnTagChange() {
 char i, tag = 0;
 if (FT800_Touch_GetTag(&tag))
 return;

 if (!(((tag>='0')&&(tag<='9'))||(tag=='_')|| ((tag >= 'a') && (tag <=
'z'))||((tag >= 'A') && (tag <= 'Z'))))
 return;

 i = strlen(EveButtonFNfilename_Caption);
 if (i >= 7) {
 return;
 }
 EveButtonFNfilename.Caption[i] = tag;
 EveButtonFNfilename.Caption[i+1] = 0;

 checkFileName();
 DrawScreen(&SettingScreen);
}

void EveButtonFNClearOnClick() {
 Strcpy(EveButtonFNfilename.Caption,"");
 EveButtonFNConfirm.Active=0;
 EveButtonFNConfirm.Color=0xAD55;
 DrawScreen(&SettingScreen);
}

void EveButtonFNBackspaceOnClick() {
 char i;
 i = strlen(EveButtonFNfilename_Caption);
 if(i==0){return;}
 EveButtonFNfilename.Caption[i-1] = 0;
 checkFileName();
 DrawScreen(&SettingScreen);
}

void EveButtonFNShiftOnClick() {
 static isShiftOn=0;
 if(isShiftOn){isShiftOn=0;EveButtonFNShift.Color=0x001F;}
 else{isShiftOn=1;EveButtonFNShift.Color=0xF800;}
 if(isShiftOn){
 EveKeys2.Caption="QWERTYUIOP";
 EveKeys3.Caption="ASDFGHJKL";
 EveKeys4.Caption="ZXCVBNM_";
 DrawScreen(&SettingScreen);
 }
 else{
 EveKeys2.Caption="qwertyuiop";
 EveKeys3.Caption="asdfghjkl";
 EveKeys4.Caption="zxcvbnm_";
 DrawScreen(&SettingScreen);
 }
}

void EveButtonFNGoBackOnClick() {
 DrawScreen(&ProfilingScreen);
}

void EveButtonSUNewRunOnClick() {
 BoxProfWaitingForCam.visible=0;
 EveTextProfWaitingForCam.visible=0;
 EveTextProfFileName.Caption="None";
 EveButtonProfStartProfiling.Color=0xAD55;
 EveButtonProfStartProfiling.Active=0;
 DrawScreen(&ProfilingScreen);
}

void EveButtonSUAboutOnClick() {
 BoxRound6.visible=1;
 EveText9.visible=1;
 EveText11.visible=1;
 EveText13.visible=1;
 EveText14.visible=1;
 EveText15.visible=1;
 EveText16.visible=1;
```

```
 EveText17.visible=1;
 EveButtonSUBack.visible=1;
 EveButtonSUBack.active=1;
 EveButtonSUNewRun.active=0;
 EveButtonSUHelp.active=0;
 EveButtonSUAbout.active=0;
 DrawScreen(&SummaryScreen);

}

void EveButtonSUHelpOnClick() {



}

void EveButtonFNConfirmOnClick() {
 EveTextProfFileName.Caption=EveButtonFNfilename_Caption;
 EveButtonProfStartProfiling.Color=0x3666;
 EveButtonProfStartProfiling.Active=1;
 DrawScreen(&ProfilingScreen);
}

void EveButtonSUBackOnClick() {
 BoxRound6.Visible=0;
 EveText9.visible=0;
 EveText11.visible=0;
 EveText13.visible=0;
 EveText14.visible=0;
 EveText15.visible=0;
 EveText16.visible=0;
 EveText17.visible=0;
 EveButtonSUBack.visible=0;
 EveButtonSUBack.active=0;
 EveButtonSUNewRun.active=1;
 EveButtonSUHelp.active=1;
 EveButtonSUAbout.active=1;
 DrawScreen(&SummaryScreen);
}

void EveButtonSmpConfigureMotorOnClick() {

 EveButtonsmpResetInc.Active=1;
 EveButtonsmpResetInc.visible=1;

 EveButtonSmpConfigureMotor.active=0;
 EveButtonSmpGotoProf.active=0;
 EveButtonSmpStopSampling.active=0;
 EveButtonSmpStartSampling.active=0;


 EveButtonSmpStopSampling.visible=0;
 EveButtonSmpStartSampling.visible=0;
 EveButtonSmpConfigureMotor.visible=0;
 EveButtonSmpGotoProf.visible=0;
 BoxRound8.visible=0;

 EveButtonsmpJogRear.active=1;
 EveButtonsmpJogFront.active=1;
 EveButtonsmpBack.active=1;

 EveButtonsmpJogRear.visible=1;
 EveButtonsmpJogFront.visible=1;
 EveButtonsmpBack.visible=1;
 BoxRound7.visible=1;
 DrawScreen(&SamplingScreen);


}

void EveButtonsmpJogRearOnPress() {
 UART1_Write(0x3E);
 Delay_ms(100);
 UART1_Write(0x3B);
}

void EveButtonsmpJogFrontOnPress() {
 UART1_Write(0x5D);
 Delay_ms(100);
 UART1_Write(0x3B);
}

void EveButtonsmpBackOnClick() {
 EveButtonsmpResetInc.Active=0;
 EveButtonsmpResetInc.visible=0;

 EveButtonSmpConfigureMotor.active=1;
 EveButtonSmpGotoProf.active=1;
 EveButtonSmpStopSampling.active=0;
 EveButtonSmpStartSampling.active=1;


 EveButtonSmpStopSampling.visible=0;
 EveButtonSmpStartSampling.visible=1;
 EveButtonSmpConfigureMotor.visible=1;
 EveButtonSmpGotoProf.visible=1;
 BoxRound8.visible=1;

 EveButtonsmpJogRear.active=0;
 EveButtonsmpJogFront.active=0;
 EveButtonsmpBack.active=0;

 EveButtonsmpJogRear.visible=0;
 EveButtonsmpJogFront.visible=0;
 EveButtonsmpBack.visible=0;
 BoxRound7.visible=0;
 DrawScreen(&SamplingScreen);
}

void EveButtonsmpResetIncOnClick() {
 UART1_Write(0xFC);
}
```

SMP.CP

```
#line 1
"C:/Users/ipm4/Desktop/PathMeT_code/2014_ian/0127/NewDesi
gnFT800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/smp.c"
#line 1
"c:/users/ipm4/desktop/pathmet_code/2014_ian/0127/newdesignft
800/newdesignft800/newdesign_code/mikroc pro for
dspic/smpfunctionheader.h"



unsigned char setBaud[]={ 0x00, 0x83, 0x84, 0x80, 0x80, 0x8C };
unsigned char setSampling[] ={ 0x00,0x83,0x89 ,0x80 ,0x80 ,0x80
,0x00 ,0x83 ,0x88 ,0x80,0x82 ,0x83};
unsigned char accumulationtime[]= { 0x00 ,0x83 ,0x8B ,0x80
,0x80,0x80 ,0x00 ,0x83 ,0x8A ,0x80 ,0x88 ,0x8C};
volatile unsigned char Laz[4];
volatile unsigned char LazIterator=0;


unsigned char
incSetGroundPitch[]={0x00,0xC1,0x00,0x00,0x00,0x00,0x00,0x3
F};
unsigned char
incSetGroundRoll[]={0x00,0xC1,0x01,0x00,0x00,0x00,0x00,0x3
E};
unsigned char
incChangeDampingTime[]={0x00,0xC6,0x00,0x02,0x63};
volatile unsigned char incXY[8];
volatile unsigned char IncIterator=0;
volatile unsigned char IncReadyBuff[8];
volatile unsigned int INCReady=0;


unsigned long REncPos=0;
unsigned long IEncPos=0;
volatile unsigned int inchWormDoneFlag=0;
volatile unsigned int inchWormMode=0;


volatile unsigned long t=0;


volatile unsigned int smpOnceflag=0;
volatile unsigned int loggingFlag=0;
volatile unsigned int nameStart=0;

volatile unsigned char measurements=0;
volatile unsigned char sendtoBufferCount=0;
volatile unsigned char bufferlistening=0;




unsigned int accx=0;
unsigned int accy=0;
unsigned int accz=0;


unsigned int pwm_period1;
unsigned long inchwormcycle=2300;


void resetAll();
void moveWorm(unsigned int direction);
void stopWorm();
void setINCGroundPlane();
```

```
#line 3
"C:/Users/ipm4/Desktop/PathMeT_code/2014_ian/0127/NewDesi
gnFT800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/smp.c"
unsigned char i=0;
void initChip(){
 PLLFBD = 68;
 CLKDIV = 0x0000;

 ANSELA = 0x00;
 ANSELB = 0x00;
 ANSELC = 0x00;
 ANSELD = 0x00;
 ANSELE = 0x00;
 ANSELG = 0x00;
}
void mapPins(){

 TRISE.B6 = 1;
 TRISE.B7 = 0;

 TRISD.B3 = 1;
 TRISD.B2 = 0;

 TRISE.B1=1;
 TRISE.B0=0;

 TRISG.B0=1;
 TRISG.B1=0;

 TRISA.B0=1;
 TRISE.B8=1;
 TRISE.B9=1;
 TRISB.B5=1;
 TRISB.B4=1;
 TRISB.B3=1;

 TRISD.B15=0;
 TRISD.B14=0;



 PPS_Mapping(86, _INPUT, _U1RX);
 PPS_Mapping(87, _OUTPUT, _U1TX);
 PPS_Mapping(67, _INPUT, _U2RX);
 PPS_Mapping(66, _OUTPUT, _U2TX);
 PPS_Mapping(81, _INPUT, _U3RX);
 PPS_Mapping(80, _OUTPUT, _U3TX);
 PPS_Mapping(112, _INPUT, _U4RX);
 PPS_Mapping(113, _OUTPUT, _U4TX);


 PPS_Mapping(16, _INPUT, _QEA1);
 PPS_Mapping(89, _INPUT, _QEB1);
 PPS_Mapping(88, _INPUT, _INDX1);
 PPS_Mapping(37, _INPUT, _QEA2);
 PPS_Mapping(35, _INPUT, _QEB2);
 PPS_Mapping(36, _INPUT, _INDX2);

 PPS_Mapping(79, _OUTPUT, _OC1);
}
void InitTimer1(){
 T1CON = 0x8010;
 T1IE_bit = 0;
 T1IF_bit = 0;
 IPC0.B14=1;
 IPC0.B13=1;
 IPC0.B12=1;
 PR1 = 8750;
}
```

```
void Timer1Interrupt() iv IVT_ADDR_T1INTERRUPT{
 T1IF_bit = 0;
 t++;
}


void initBufferchipUART(){

UART1_Init_Advanced(921600,_UART_8BIT_NOPARITY,_UA
RT_ONE_STOPBIT,_UART_HI_SPEED);
 Delay_ms(100);
 IFS0.B11=0;
 U1STA.OERR=0;
 IPC2.B14=1;
 IPC2.B13=0;
 IPC2.B12=0;
 IEC0.B11=1;
}
void BufferChipInterrupt() iv IVT_ADDR_U1RXINTERRUPT {
 unsigned char temp = 0;
 IFS0.B11=0;
 temp=UART1_Read();
 if(temp==0xF3){
 loggingFlag=0;

 }
 if(temp==0xE7){
 bufferlistening=1;
 Delay_ms(100);
 REncPos=0;
 POS1HLD=0;
 POS1CNTL=0;
 if(inchWormMode){moveWorm(0);}
 T1IE_bit=1;
 }
 if(temp==0xA9){
 bufferlistening=0;
 }

}

void initBufferchipSPI(){
 SPI2_Init_Advanced(_SPI_MASTER, _SPI_8_BIT,
_SPI_PRESCALE_SEC_4, _SPI_PRESCALE_PRI_4,
_SPI_SS_DISABLE, _SPI_DATA_SAMPLE_MIDDLE,
_SPI_CLK_IDLE_LOW, _SPI_ACTIVE_2_IDLE);
 Delay_ms(100);
}

void initScreenChipCommunication(){


UART2_Init_Advanced(921600,_UART_8BIT_NOPARITY,_UA
RT_ONE_STOPBIT,_UART_HI_SPEED);
 Delay_ms(100);
 IFS1.B14=0;
 U2STA.OERR=0;
 IPC7.B10=1;
 IPC7.B9=0;
 IPC7.B8=1;
 IEC1.B14=1;
}
void ScreenChipInterrupt() iv IVT_ADDR_U2RXINTERRUPT {
 unsigned char temp = 0;
 IFS1.B14=0;
 temp=UART2_Read();
 if(nameStart){
 if(temp==0xB5){
 nameStart=0;
 return;
 }
 UART1_Write(temp);
 }
 else{
 switch (temp){
 case 0x31:
 smpOnceflag=1;
 break;
 case 0xA2:
 if(loggingflag){return;}
 UART1_Write(0xD3);
 loggingFlag=1;

 sendtoBufferCount=0;
 measurements=0;
 t=0;

 break;
 case 0xB7:
 loggingFlag=0;

 break;
 case 0xE3:
 nameStart=1;
 UART1_Write(0xF3);
 break;
 case 0x1C:
 inchWormMode=1;
 IEncPos=0;
 POS2HLD=0;
 POS2CNTL=0;


 break;
 case 0x5D:
 moveWorm(1);
 break;
 case 0x3E:
 moveWorm(0);
 break;
 case 0x3B:
 stopWorm();
 break;

 case 0xFC:
 setINCGroundPlane();
 break;
 }
 }

}


void initLaz(){
 unsigned int iterator=0;


UART3_Init_Advanced(9600,_UART_8BIT_EVENPARITY,_U
ART_ONE_STOPBIT,_UART_HI_SPEED);
 Delay_ms(100);
 for
(iterator=0;iterator<7;iterator++){UART3_Write(setBaud[iterator]
);}


UART3_Init_Advanced(460800,_UART_8BIT_EVENPARITY,_
UART_ONE_STOPBIT,_UART_HI_SPEED);
 Delay_ms(100);
```

```
 for
(iterator=0;iterator<13;iterator++){UART3_Write(setSampling[ite
rator]);}
 Delay_ms(10);
 for
(iterator=0;iterator<13;iterator++){UART3_Write(accumulationti
me[iterator]);}
 Delay_ms(10);


 U3STA.OERR=0;

 U3STA.B7=0;
 IPC20.B10=1;
 IPC20.B9=0;
 IPC20.B8=1;
 U3STA.B7=0;
 IFS5.B2=0;
 IEC5.B2=1;

 Laz[3]=0;
 Laz[2]=0;
 Laz[1]=0;Laz[0]=0;
 LazIterator=0;
}
void LazInterrupt() iv IVT_ADDR_U3RXINTERRUPT ics
ICS_AUTO {
 IFS5.B2=0;
 Laz[LazIterator++]=U3RXREG&0b00001111;
 if(LazIterator==4){LazIterator=0; }
}
void initInc(){
 unsigned int initializer=0;

for(initializer=0;initializer<8;initializer++){IncReadyBuff[initializ
er]=initializer;}
 UART4_Init(115200);
 Delay_ms(500);
 IFS5.B8=0;
 U4STA.B7=0;
 IPC22.B2=1;
 IPC22.B1=1;
 IPC22.B0=0;
 IEC5.B8=1;
 U4STA.OERR=0;

for(initializer=0;initializer<sizeof(incChangeDampingTime);initial
izer++){
 UART4_Write(incChangeDampingTime[initializer]);
 }

 Delay_ms(500);
 IncIterator=0;

}
void IncInterrupt() iv IVT_ADDR_U4RXINTERRUPT{
 IFS5.B8=0;
 if(IncIterator>=8){UART4_Read();IncIterator++;}
 else{incXY[IncIterator++]=U4RXREG;}

if(IncIterator==15){IncIterator=0;memmove(IncReadyBuff,incXY
,8);}

}
void setINCGroundPlane(){
 unsigned int i=0;

 for(i=0;i<8;i++){
 UART4_Write(incSetGroundPitch[i]);
 }
 Delay_ms(20);
 for(i=0;i<8;i++){
 UART4_Write(incSetGroundRoll[i]);
 }
 Delay_ms(500);
 IncIterator=0;

}
void initAcc(){
 ANSELE.ANSE2=1;
 ANSELE.ANSE3=1;
 ANSELE.ANSE4=1;
 TRISE.B4=1;
 TRISE.B3=1;
 TRISE.B2=1;
 ADC1_Init();
}
void initRollingQEI(){
 QEI1CON=0b1000000000000000;
}
void initInchWormQEI(){
 QEI2CON=0b1000000000000000;
}
void initMotor(){
 initInchWormQEI();
 pwm_period1 = PWM_Init(500, 1, 1, 2);
}
void initAll(){

 initChip();
 mapPins();
 TRISF.B4=0;
 LATF.B4=1;
 initTimer1();
 initBufferchipUART();
 initBufferchipSPI();
 initScreenChipCommunication();

 Delay_ms(20000);


 initAcc();
 initLaz();
 initInc();
 initRollingQEI();

 initMotor();

}
void resetAll(){
 T1IE_bit= 0;
 t=0;
 U1STA.OERR=0;
 U2STA.OERR=0;
 U3STA.OERR=0;
 U4STA.OERR=0;
 UART1_Read();
 UART2_Read();
 UART3_Read();
 UART4_Read();
 LazIterator=0;
 IncIterator=0;
 REncPos=0;
 POS1HLD=0;
 POS1CNTL=0;
 sendtoBufferCount=0;
}


void getRQEIPOS(unsigned long *pos){
```

```
 static unsigned int posL;
 static unsigned long posH;
 posL=POS1CNTL;
 posH=POS1HLD;
 *pos=posL|(posH<<16);
}
void getIQEIPOS(unsigned long *pos){
 static unsigned int posL;
 static unsigned long posH;
 posL=POS2CNTL;
 posH=POS2HLD;
 *pos=posL|(posH<<16);
}
void getLaz(){
 UART3_Write(0x00);UART3_Write(0x86);
}
void getXYZAcc(unsigned int *outputx,unsigned int
*outputy,unsigned int *outputz){
 *outputx=ADC1_Get_Sample(26);
 *outputy=ADC1_Get_Sample(27);
 *outputz=ADC1_Get_Sample(28);

}
void getAllAngles(){
 UART4_Write(0x00);UART4_Write(0xE1);
}

void sampleOnce(){
 getLaz();
 if(inchWormMode){
 getIQEIPOS(&IEncPos);
 }
 else{
 getRQEIPOS(&REncPos);
 }
 measurements++;
 if(measurements==1){
 getAllAngles();
 getXYZAcc(&accx,&accy,&accz);
 Delay_us(300);
 }
 else if(measurements==8){
 measurements=0;
 Delay_us(400);
 }
 else{
 Delay_us(400);
 }


}
void sendToBuffer(){
 volatile unsigned long tempT;
 volatile unsigned long tempEncPos;

 if(inchWormMode){
 tempEncPos=IEncPos;
 }
 else{
 tempEncPos=REncPos;
 }

 T1IE_bit = 0;
 tempT=t;
 T1IE_bit = 1;



SPI2_Write(Laz[3]);SPI2_Write(Laz[2]);SPI2_Write(Laz[1]);SPI2
_Write(Laz[0]);


SPI2_Write(tempEncPos>>24);SPI2_Write(tempEncPos>>16);SP
I2_Write(tempEncPos>>8);SPI2_Write(tempEncPos);


SPI2_Write(tempT>>16);SPI2_Write(tempT>>8);SPI2_Write(tem
pT);
 sendtoBufferCount++;


 if(sendtoBufferCount==8){
 sendtoBufferCount=0;


SPI2_Write(IncReadyBuff[0]);SPI2_Write(IncReadyBuff[1]);SPI2
_Write(IncReadyBuff[2]);SPI2_Write(IncReadyBuff[3]);


SPI2_Write(IncReadyBuff[4]);SPI2_Write(IncReadyBuff[5]);SPI2
_Write(IncReadyBuff[6]);SPI2_Write(IncReadyBuff[7]);

 SPI2_Write(accx>>8);SPI2_Write(accx);

 SPI2_Write(accy>>8);SPI2_Write(accy);

 SPI2_Write(accz>>8);SPI2_Write(accz);
 }
}
void smpsendToScreen(){
 volatile unsigned long tempEncPos;



 tempEncPos=REncPos;


UART2_Write(Laz[3]);UART2_Write(Laz[2]);UART2_Write(La
z[1]);UART2_Write(Laz[0]);


UART2_Write(tempEncPos>>24);UART2_Write(tempEncPos>>
16);UART2_Write(tempEncPos>>8);UART2_Write(tempEncPos)
;

UART2_Write(IncReadyBuff[0]);UART2_Write(IncReadyBuff[1]
);UART2_Write(IncReadyBuff[2]);UART2_Write(IncReadyBuff[
3]);

UART2_Write(IncReadyBuff[4]);UART2_Write(IncReadyBuff[5]
);UART2_Write(IncReadyBuff[6]);UART2_Write(IncReadyBuff[
7]);
 UART2_Write(accx>>8);UART2_Write(accx);
 UART2_Write(accy>>8);UART2_Write(accy);
 UART2_Write(accz>>8);UART2_Write(accz);
}
void logsendToScreen(){
 static unsigned int temp=0;
 unsigned long tempT;
 unsigned long tempEnc;
 tempT=t;
 if(inchWormMode){
 tempEnc=IEncPos;
 }
 else{
 tempEnc=REncPos;
 }
 temp++;
 if(temp==100){
```

```
 temp=0;

UART2_Write(Laz[3]);UART2_Write(Laz[2]);UART2_Write(La
z[1]);UART2_Write(Laz[0]);

UART2_Write(IncReadyBuff[0]);UART2_Write(IncReadyBuff[1]
);UART2_Write(IncReadyBuff[2]);UART2_Write(IncReadyBuff[
3]);

UART2_Write(IncReadyBuff[4]);UART2_Write(IncReadyBuff[5]
);UART2_Write(IncReadyBuff[6]);UART2_Write(IncReadyBuff[
7]);

UART2_Write(tempT>>16);UART2_Write(tempT>>8);UART2_
Write(tempT);

UART2_Write(tempEnc>>24);UART2_Write(tempEnc>>16);UA
RT2_Write(tempEnc>>8);UART2_Write(tempEnc);
 }
}
void moveWorm(unsigned int direction){
 PWM_Start(1);
 if(direction==1){
 LATD.B14=1;
 }
 else{
 LATD.B14=0;
 }
 PWM_Set_Duty(pwm_period1/2, 1);
}
void stopWorm(){
 PWM_Stop(1);
}
void main() {
 unsigned char temp;
 initAll();
 LATF.B4=0;
 Delay_ms(100);
 resetAll();
 while(1){
 if(smpOnceflag){
 smpOnceflag=0;
 LATF.B4=1;
 sampleOnce();
 Delay_ms(10);
 smpsendToScreen();
 LATF.B4=0;
 Delay_ms(5);
 }
 if(sendtoBufferCount!=0){ LATF.B4=~LATF.B4;}
 while((loggingFlag+bufferlistening)==2){
 sampleOnce();

 sendToBuffer();

 logsendToScreen();

if(inchWormMode){if(IEncPos>=inchwormcycle){stopWorm();D
elay_ms(500);PWM_Start(1);loggingFlag=0;moveWorm(1);inch
WormDoneFlag=1;}}
 if(!loggingFlag){
 Delay_ms(1000);
 UART1_Write(0xE2);
 resetAll();
 Delay_ms(1000);
 UART2_Write(0xAA);
 Delay_us(10);
 UART2_Write(0xAA);
 Delay_us(10);
 UART2_Write(0xAA);
 Delay_us(10);
 UART2_Write(0xAA);
 Delay_us(10);
 UART2_Write(0xAA);
 Delay_us(10);
 UART2_Write(0xAA);

 measurements=0;
 sendtoBufferCount=0;
 T1IE_bit= 0;
 t=0;
 while(inchWormDoneFlag){
 LATF.B4=1;
 getIQEIPOS(&IEncPos);
 if(IEncPos<=2){
 stopWorm();
 inchWormDoneFlag=0;
 LATF.B4=0;
 inchWormMode=0;
 }
 }
 break;
 }

 }
 }
}
```

NEWDESIGN_MAIN.CP

```
#line 1
"C:/Users/ipm4/Desktop/latest_code_01062014/0121/NewDesignF
T800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/NewDesign_main.c"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_objects.h"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
#line 21
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
typedef unsigned char uint8_ft8;
typedef signed char int8_ft8;
typedef unsigned int uint16_ft8;
typedef int int16_ft8;
typedef unsigned long uint32_ft8;
typedef long int32_ft8;
typedef unsigned long long uint64_ft8;
typedef long long int64_ft8;

typedef enum {cmdREAD, cmdWRITE} TCmd;

typedef enum {opNONE, opREAD, opWRITE} TOpr;

typedef enum {umNONE, umDL=1<<0, umCP=1<<1,
umGRAM=1<<2} TUpdateMode;

typedef uint8_ft8 TFT800Alpha;
typedef uint8_ft8 TFT800Handle;

typedef struct
{
 int16_ft8 X;
 int16_ft8 Y;
} TFT800Point;

typedef struct
{
 int16_ft8 Left;
 int16_ft8 Top;
 uint16_ft8 Width;
 uint16_ft8 Height;
} TFT800Rect;

typedef union
{
 struct
 {
 uint8_ft8 B;
 uint8_ft8 G;
 uint8_ft8 R;
 };
 uint32_ft8 RGB;
} TFT800Color;

typedef struct
{
 TFT800Color Color;
 TFT800Alpha Alpha;
 uint8_ft8 Width;
} TFT800Pen;

typedef struct
{
 TFT800Color ColorBg;
 TFT800Color Color;


 TFT800Color ColorTo;
 uint8_ft8 Gradient;
 TFT800Alpha Alpha;
 uint8_ft8 Style;
} TFT800Brush;



typedef struct
{

 uint8_ft8 FontWidth[ 256 ];
 uint32_ft8 FontBitmapFormat;
 uint32_ft8 FontLineStride;
 uint32_ft8 FontWidthInPixels;
 uint32_ft8 FontHeightInPixels;
 uint32_ft8 PointerToFontGraphicsData;
} TFT800FontGraphics;

typedef struct
{
 TFT800FontGraphics GrData;
 uint16_ft8 FirstChar;
 uint16_ft8 LastChar;
} TFT800CharSetDsc;

typedef struct
{
 uint32_ft8 Name;
 uint32_ft8 Source;
 TFT800Color Color;
 TFT800Alpha Alpha;
 TFT800Handle Handle;
 TFT800CharSetDsc CharSetDsc;
} TFT800Font;

typedef struct
{
 uint8_ft8 Format;
 uint16_ft8 LineStride;
 uint16_ft8 Height;
} TFT800BmpLayout;

typedef struct
{
 uint8_ft8 FwFlags;
 uint16_ft8 Width;
 uint16_ft8 Height;
} TFT800BmpSize;

typedef struct
{
 uint32_ft8 Source;
 TFT800BmpLayout Layout;
 TFT800BmpSize Size;
} TFT800BmpConfig;

typedef struct
{
 TFT800BmpConfig Configs[16];
 uint8_ft8 Current;
} TFT800BmpHandle;

typedef struct
```

```
{
 TFT800BmpConfig Config;
 TFT800Color BlendColor;
 TFT800Alpha Alpha;
 TFT800Handle Handle;
} TFT800Bitmap;

typedef struct
{
 uint16_ft8 Hour;
 uint16_ft8 Min;
 uint16_ft8 Sec;
 uint16_ft8 mSec;
} TFT800Time;

typedef struct
{
 uint32_ft8 DrawingOptions;
 uint16_ft8 Val;
 uint16_ft8 Size;
 uint16_ft8 Range;
 uint16_ft8 Minor;
 uint16_ft8 Major;
 TFT800Time Time;
 uint8_ft8 Style;
} TFT800CoProcGraphics;

typedef struct
{
 TFT800Pen Pen;
 TFT800Brush Brush;
 TFT800Font Font;
 TFT800Bitmap Bitmap;

 TFT800Rect ClipRect;
 TFT800Point Cursor;

 uint8_ft8 Stencil;
 uint8_ft8 Tag;

 TFT800CoProcGraphics CPGraphics;
} TFT800Canvas;

typedef struct
{
 TOpr Opr;
 uint32_ft8 RWPtr;
} TFT800IO;


typedef struct
{

 uint32_ft8 Frequency;
 uint32_ft8 OutRenderMode;
 uint32_ft8 RenderReadScanLine;
 uint32_ft8 RenderWriteTrigger;

 uint32_ft8 hCycle;
 uint32_ft8 hOffset;
 uint32_ft8 hSize;
 uint32_ft8 hSync0;
 uint32_ft8 hSync1;

 uint32_ft8 vCycle;
 uint32_ft8 vOffset;
 uint32_ft8 vSize;
 uint32_ft8 vSync0;
 uint32_ft8 vSync1;
 uint32_ft8 Rotate180;
 uint32_ft8 OutBits;
 uint32_ft8 OutDither;
 uint32_ft8 OutSwizzle;
 uint32_ft8 OutCSpread;
 uint32_ft8 PClockPolarity;
 uint32_ft8 PClock;
} TFT800Display;

typedef struct
{
 uint32_ft8 TouchMode;
 uint32_ft8 TouchADCMode;
 uint32_ft8 TouchCharge;
 uint32_ft8 TouchSettle;
 uint32_ft8 TouchOversample;
 uint32_ft8 TouchRZThreshold;
} TFT800Touch;

typedef struct
{
 uint32_ft8 TransformA;
 uint32_ft8 TransformB;
 uint32_ft8 TransformC;
 uint32_ft8 TransformD;
 uint32_ft8 TransformE;
 uint32_ft8 TransformF;
} TFT800TouchTransform;

typedef struct
{
 uint8_ft8 Enable;
 uint8_ft8 Mask;
 uint8_ft8 Flags;
} TFT800Interrupt;

typedef struct
{
 uint8_ft8 Effect;
 uint8_ft8 Pitch;
 uint8_ft8 Volume;
 uint8_ft8 Play;
} TFT800Sound;

typedef struct
{
 uint32_ft8 StartAddress;
 uint32_ft8 Length;
 uint16_ft8 Frequency;
 uint8_ft8 Format;
 uint8_ft8 Loop;
 uint8_ft8 Volume;
 uint8_ft8 Play;
} TFT800Audio;

typedef struct
{

 uint16_ft8 Freq;
 uint8_ft8 Duty;
} TFT800PWM;

typedef struct
{
 uint8_ft8 GPIODIR;
 uint8_ft8 GPIO;
} TFT800GPIO;

typedef struct
{
```

```
 TFT800PWM *pPWMCfg;
 TFT800GPIO *pGPIOCfg;
 TFT800Audio *pAudioCfg;
 TFT800Sound *pSoundCfg;
 TFT800Touch *pTouchCfg;
 TFT800Display *pDisplayCfg;
 TFT800Interrupt *pInterruptCfg;
 TFT800TouchTransform *pTTransformCfg;
} TFT800Config;



typedef struct
{
 struct
 {




 TFT800Color Color;
 TFT800Alpha Alpha;

 uint8_ft8 Tag;

 TFT800Color ClearColor;
 TFT800Alpha ClearAlpha;
 uint8_ft8 ClearStencil;
 uint8_ft8 ClearTag;

 uint16_ft8 LineWidth;
 uint16_ft8 PointSize;

 uint16_ft8 ScissorLeft;
 uint16_ft8 ScissorTop;
 uint16_ft8 ScissorWidth;
 uint16_ft8 ScissorHeight;

 uint8_ft8 BmpHandle;
 uint8_ft8 Cell;

 struct
 {
 Color : 1;
 Alpha : 1;
 Tag : 1;
 ClearColor : 1;
 ClearAlpha : 1;
 ClearStencil : 1;
 ClearTag : 1;
 LineWidth : 1;
 PointSize : 1;
 ScissorPos : 1;
 ScissorSize : 1;
 BmpHandle : 1;
 Cell : 1;

 Unused : 32-13;

 } UpdateFlags;


 } Context;

 TFT800BmpConfig BmpHandleCfg[16];

 int8_ft8 CurrGrPrim;
} TFT800Graphics;

typedef struct
{
 TFT800IO IO;

 TFT800Sound Sound;

 TFT800Audio Audio;

 TFT800Graphics Graphics;

 struct
 { uint16_ft8 Width;
 uint16_ft8 Height;
 } Display;

 uint8_ft8 UpdateMode;

} TFT800Controller;
#line 6
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_objects.h"
typedef enum {taNone, taLeft, taCenter, taRight, taCenterX,
taCenterY, taRightX} TTextAlign;

typedef struct Screen TScreen;

typedef unsigned long TPointer;

typedef struct tagObjInfo {
 TPointer Obj;
 char Type;
 char Order;
 char Flags;

 char HitTag;
 int HitX;
 int HitY;
} TObjInfo;

typedef struct tagTouchStat {
 char Pressed;

 char Tag;
 int X;
 int Y;

 TObjInfo ActObjInfo;
} TTouchStat;

typedef void (*TDrawHandler)(TPointer aObj);

typedef void (*TEvtAction)();

typedef struct tagEvtSound {
 char SndAct;
 char Effect;
 char Pitch;
 char Volume;
} TEvtSound;

typedef const far struct tagCEvent {
 TEvtAction Action;
 TEvtSound Sound;
} TCEvent;

typedef struct tagEvent {
 TEvtAction Action;
 TEvtSound Sound;
} TEvent;
```

```
typedef const far struct tagCRect {
 int Left;
 int Top;
 int Width;
 int Height;
} TCRect;

typedef struct tagRect {
 int Left;
 int Top;
 int Width;
 int Height;
} TRect;

typedef struct tagBox {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char Gradient;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TBox;

typedef far const code struct tagCBox {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char Gradient;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCBox;

typedef struct tagBox_Round {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Corner_Radius;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TBox_Round;

typedef far const code struct tagCBox_Round {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Corner_Radius;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCBox_Round;

typedef struct tagLine {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int First_Point_X;
 int First_Point_Y;
 int Second_Point_X;
 int Second_Point_Y;
 char Pen_Width;
 unsigned int Pen_Color;
} TLine;

typedef struct tagEveGauge {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Radius;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Major;
 char Minor;
 unsigned int Value;
 unsigned int Range;
 char Flat;
 char NoBackground;
 char NoPointer;
```

```
 char TicksVisible;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveGauge;

typedef struct tagEveKeys {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char *Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Flat;
 char AutoSize;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveKeys;

typedef struct tagEveProgressBar {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Background_Color;
 unsigned int Color;
 unsigned int Value;
 unsigned int Range;
 char Flat;
} TEveProgressBar;

typedef struct tagEveToggle {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Background_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char *StateOFF_Caption;
 char *StateON_Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 unsigned int State;
 char Flat;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveToggle;

typedef struct tagEveButton {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char *Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Flat;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveButton;

typedef struct tagEveText {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char *Caption;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveText;

typedef far const code struct tagCEveText {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
```

```
 int Top;
 int Width;
 int Height;
 far const code char *Caption;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCEveText;

typedef far const code struct tagCEveNumber {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Text_Length;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 signed long Value;
 unsigned char Signed;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCEveNumber;

struct Screen {
 unsigned int Color;
 unsigned int Width;
 unsigned int Height;
 unsigned short ObjectsCount;
 unsigned short BoxesCount;
 TBox *far const code *Boxes;
 unsigned short CBoxesCount;
 TCBox *far const code *CBoxes;
 unsigned short Boxes_RoundCount;
 TBox_Round *far const code *Boxes_Round;
 unsigned short CBoxes_RoundCount;
 TCBox_Round *far const code *CBoxes_Round;
 unsigned short LinesCount;
 TLine *far const code *Lines;
 unsigned short EveGaugesCount;
 TEveGauge *far const code *EveGauges;
 unsigned short EveKeysCount;
 TEveKeys *far const code *EveKeys;
 unsigned short EveProgressBarsCount;
 TEveProgressBar *far const code *EveProgressBars;
 unsigned short EveTogglesCount;
 TEveToggle *far const code *EveToggles;
 unsigned short EveButtonsCount;
 TEveButton *far const code *EveButtons;
 unsigned short EveTextsCount;
 TEveText *far const code *EveTexts;
 unsigned short CEveTextsCount;
 TCEveText *far const code *CEveTexts;
 unsigned short CEveNumbersCount;
 TCEveNumber *far const code *CEveNumbers;
 unsigned long DynResStart;
 unsigned short Active;
 unsigned short SniffObjectEvents;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnTagChange;
 TEvent *OnPress;
};

extern const VTFT_OT_BOX;
extern const VTFT_OT_CBOX;
extern const VTFT_OT_BOXROUND;
extern const VTFT_OT_CBOXROUND;
extern const VTFT_OT_LINE;
extern const VTFT_OT_EVEGAUGE;
extern const VTFT_OT_EVEKEYS;
extern const VTFT_OT_EVEPROGRESSBAR;
extern const VTFT_OT_EVETOGGLE;
extern const VTFT_OT_EVEBUTTON;
extern const VTFT_OT_EVETEXT;
extern const VTFT_OT_CEVETEXT;
extern const VTFT_OT_CEVENUMBER;

extern const VTFT_EVT_UP;
extern const VTFT_EVT_DOWN;
extern const VTFT_EVT_CLICK;
extern const VTFT_EVT_PRESS;

extern const VTFT_SNDACT_NONE;
extern const VTFT_SNDACT_PLAY;
extern const VTFT_SNDACT_STOP;

extern const VTFT_LOAD_RES_NONE;
extern const VTFT_LOAD_RES_STATIC;
extern const VTFT_LOAD_RES_DYNAMIC;
extern const VTFT_LOAD_RES_ALL;

extern const VTFT_DISPLAY_EFF_NONE;
extern const VTFT_DISPLAY_EFF_LIGHTS_FADE;
extern const VTFT_DISPLAY_EFF_LIGHTS_OFF;

extern const VTFT_INT_REPAINT_ON_DOWN;
extern const VTFT_INT_REPAINT_ON_UP;
extern const VTFT_INT_BRING_TO_FRONT;
extern const VTFT_INT_INTRINSIC_CLICK_EFF;

extern const TPointer DrawHandlerTable[44];
```

```
extern const TFT800PWM VTFT_FT800_CONFIG_PWM;
extern const TFT800GPIO VTFT_FT800_CONFIG_GPIO;
extern const TFT800Sound VTFT_FT800_CONFIG_SOUND;
extern const TFT800Audio VTFT_FT800_CONFIG_AUDIO;
extern const TFT800Display VTFT_FT800_CONFIG_DISPLAY;
extern const TFT800Interrupt
VTFT_FT800_CONFIG_INTERRUPT;
extern const TFT800Touch VTFT_FT800_CONFIG_TOUCH;
extern const TFT800TouchTransform
VTFT_FT800_CONFIG_TOUCHTRANSFORM;

extern TTouchStat TouchS;

extern TScreen SplashScreen;

extern TCEveText EveTextSplashScreenName;
extern TEveButton EveButtonSplashInit;
extern TEvent EveButtonSplashInit_OnClick;
extern TEveText EveTextSplashScreenStatus;
extern TEveProgressBar EveProgressBar1;

extern TEveProgressBar *far const code
SplashScreen_EveProgressBars[1];
extern TEveButton *far const code SplashScreen_EveButtons[1];
extern TEveText *far const code SplashScreen_EveTexts[1];
extern TCEveText *far const code SplashScreen_CEveTexts[1];

extern TScreen SamplingScreen;

extern TCBox_Round BoxRoundsmpScn1;
extern TCEveText EveTextSampScnLabel1;
extern TCEveText EveTextsampscnlabel2;
extern TCEveText EveTextsampscnlabel4;
extern TCEveText EveTextsampscnlabel3;
extern TCEveText EveTextsampscnlabel5;
extern TCEveText EveTextsampscnlabel6;
extern TCBox_Round BoxRoundSmpscn2;
extern TEveText EveTextEncoderSamp;
extern TEveText EveTextLaserSmp;
extern TEveText EveTextPitchSmp;
extern TEveText EveTextGPSSmp;
extern TEveText EveTextSDSmp;
extern TEveText EveTextRollSmp;
extern TEveText EveTextAccx;
extern TEveText EveTextAccy;
extern TEveText EveTextAccz;
extern TBox_Round BoxRound7;
extern TEveButton EveButtonsmpJogRear;
extern TEvent EveButtonsmpJogRear_OnPress;
extern TEveButton EveButtonsmpJogFront;
extern TEvent EveButtonsmpJogFront_OnPress;
extern TEveButton EveButtonsmpBack;
extern TEvent EveButtonsmpBack_OnClick;
extern TBox_Round BoxRound8;
extern TEveButton EveButtonSmpStartSampling;
extern TEvent EveButtonSmpStartSampling_OnClick;
extern TEveButton EveButtonSmpStopSampling;
extern TEvent EveButtonSmpStopSampling_OnClick;
extern TEveButton EveButtonSmpConfigureMotor;
extern TEvent EveButtonSmpConfigureMotor_OnClick;
extern TEveButton EveButtonSmpGotoProf;
extern TEvent EveButtonSmpGotoProf_OnClick;
extern TEveButton EveButtonsmpResetInc;
extern TEvent EveButtonsmpResetInc_OnClick;

extern TBox_Round *far const code
SamplingScreen_Boxes_Round[2];
extern TCBox_Round *far const code
SamplingScreen_CBoxes_Round[2];
extern TEveButton *far const code
SamplingScreen_EveButtons[8];
extern TEveText *far const code SamplingScreen_EveTexts[9];
extern TCEveText *far const code SamplingScreen_CEveTexts[6];

extern TScreen ProfilingScreen;

extern TCBox Box1;
extern TCBox Box2;
extern TCEveText EveTextProflabel1;
extern TCEveText EveTextProflabel2;
extern TCEveNumber EveNumber1;
extern TCEveNumber EveNumber2;
extern TCEveNumber EveNumber3;
extern TCEveNumber EveNumber4;
extern TCEveNumber EveNumber5;
extern TCEveNumber EveNumber6;
extern TCBox_Round BoxRound1;
extern TEveGauge EveGaugeProfSpeed;
extern TEveGauge EveGaugeProfCrossSlp;
extern TCEveText EveTextProflabel5;
extern TCEveText EveTextProflabel4;
extern TEveToggle EveToggleProfMode;
extern TEvent EveToggleProfMode_OnClick;
extern TCEveText EveTextProflabel3;
extern TEveButton EveButtonProfStartProfiling;
extern TEvent EveButtonProfStartProfiling_OnClick;
extern TEveButton EveButtonProfStopProfiling;
extern TEvent EveButtonProfStopProfiling_OnClick;
extern TLine Line1;
extern TLine Line2;
extern TLine Line3;
extern TLine Line4;
extern TLine Line5;
extern TLine Line6;
extern TLine Line7;
extern TLine Line8;
extern TLine Line9;
extern TLine Line10;
extern TLine Line11;
extern TLine Line12;
extern TLine Line13;
extern TLine Line14;
extern TLine Line15;
extern TLine Line16;
extern TLine Line17;
extern TLine Line18;
extern TLine Line19;
extern TLine Line20;
extern TLine Line21;
extern TLine Line22;
extern TLine Line23;
extern TLine Line24;
extern TLine Line25;
extern TLine Line26;
extern TLine Line27;
extern TLine Line28;
extern TLine Line29;
extern TLine Line30;
extern TLine Line31;
extern TLine Line32;
extern TLine Line33;
extern TLine Line34;
extern TLine Line35;
extern TLine Line36;
extern TLine Line37;
extern TLine Line38;
extern TLine Line39;
```

```
extern TLine Line40;
extern TLine Line41;
extern TLine Line42;
extern TLine Line43;
extern TLine Line44;
extern TLine Line45;
extern TLine Line46;
extern TLine Line47;
extern TLine Line48;
extern TLine Line49;
extern TLine Line50;
extern TEveText EveText1;
extern TEveButton EveButtonProfCreateFile;
extern TEvent EveButtonProfCreateFile_OnClick;
extern TCEveText EveText10;
extern TEveText EveTextProfFileName;
extern TCEveText EveText12;
extern TBox BoxProfWaitingForCam;
extern TEveText EveTextProfWaitingForCam;

extern TBox *far const code ProfilingScreen_Boxes[1];
extern TCBox *far const code ProfilingScreen_CBoxes[2];
extern TCBox_Round *far const code
ProfilingScreen_CBoxes_Round[1];
extern TLine *far const code ProfilingScreen_Lines[50];
extern TEveGauge *far const code ProfilingScreen_EveGauges[2];
extern TEveToggle *far const code
ProfilingScreen_EveToggles[1];
extern TEveButton *far const code
ProfilingScreen_EveButtons[3];
extern TEveText *far const code ProfilingScreen_EveTexts[3];
extern TCEveText *far const code ProfilingScreen_CEveTexts[7];
extern TCEveNumber *far const code
ProfilingScreen_CEveNumbers[6];

extern TScreen SettingScreen;
extern TEvent SettingScreen_OnTagChange;

extern TEveButton EveButtonFNfilename;
extern TCBox_Round BoxRound2;
extern TEveKeys EveKeys1;
extern TEveKeys EveKeys2;
extern TEveKeys EveKeys3;
extern TEveKeys EveKeys4;
extern TCEveText EveText2;
extern TCBox_Round BoxRound3;
extern TEveButton EveButtonFNBackspace;
extern TEvent EveButtonFNBackspace_OnClick;
extern TEveButton EveButtonFNClear;
extern TEvent EveButtonFNClear_OnClick;
extern TEveButton EveButtonFNConfirm;
extern TEvent EveButtonFNConfirm_OnClick;
extern TEveButton EveButtonFNGoBack;
extern TEvent EveButtonFNGoBack_OnClick;
extern TEveButton EveButtonFNShift;
extern TEvent EveButtonFNShift_OnClick;

extern TCBox_Round *far const code
SettingScreen_CBoxes_Round[2];
extern TEveKeys *far const code SettingScreen_EveKeys[4];
extern TEveButton *far const code SettingScreen_EveButtons[6];
extern TCEveText *far const code SettingScreen_CEveTexts[1];

extern TScreen SummaryScreen;

extern TCBox_Round BoxRound4;
extern TCEveText EveText3;
extern TCEveText EveText4;
extern TCEveText EveText5;
extern TCEveText EveText6;
extern TCEveText EveText7;
extern TCBox_Round BoxRound5;
extern TEveButton EveButtonSUNewRun;
extern TEvent EveButtonSUNewRun_OnClick;
extern TEveButton EveButtonSUHelp;
extern TEvent EveButtonSUHelp_OnClick;
extern TEveButton EveButtonSUAbout;
extern TEvent EveButtonSUAbout_OnClick;
extern TEveText EveTextSUtime;
extern TEveText EveTextSUdistance;
extern TEveText EveTextSUSpeed;
extern TEveText EveTextSUFilename;
extern TEveText EveTextSUAverageSpeed;
extern TEveText EveText8;
extern TBox_Round BoxRound6;
extern TEveText EveText9;
extern TEveText EveText11;
extern TEveText EveText13;
extern TEveText EveText14;
extern TEveText EveText15;
extern TEveText EveText16;
extern TEveText EveText17;
extern TEveButton EveButtonSUBack;
extern TEvent EveButtonSUBack_OnClick;

extern TBox_Round *far const code
SummaryScreen_Boxes_Round[1];
extern TCBox_Round *far const code
SummaryScreen_CBoxes_Round[2];
extern TEveButton *far const code
SummaryScreen_EveButtons[4];
extern TEveText *far const code SummaryScreen_EveTexts[13];
extern TCEveText *far const code
SummaryScreen_CEveTexts[5];

extern TScreen *CurrentScreen;



void EveButtonFNBackspaceOnClick();
void EveButtonFNClearOnClick();
void EveButtonFNConfirmOnClick();
void EveButtonFNGoBackOnClick();
void EveButtonFNShiftOnClick();
void EveButtonProfCreateFileOnClick();
void EveButtonProfStartProfilingOnClick();
void EveButtonProfStopProfilingOnClick();
void EveButtonsmpBackOnClick();
void EveButtonSmpConfigureMotorOnClick();
void EveButtonSmpGotoProfOnClick();
void EveButtonsmpJogFrontOnPress();
void EveButtonsmpJogRearOnPress();
void EveButtonsmpResetIncOnClick();
void EveButtonSmpStartSamplingOnClick();
void EveButtonSmpStopSamplingOnClick();
void EveButtonSplashInitOnClick();
void EveButtonSUAboutOnClick();
void EveButtonSUBackOnClick();
void EveButtonSUHelpOnClick();
void EveButtonSUNewRunOnClick();
void EveToggleProfModeOnClick();
void SettingScreenOnTagChange();



extern const code far char EveTextSplashScreenName_Caption[];
extern char EveButtonSplashInit_Caption[];
extern char EveTextSplashScreenStatus_Caption[];
extern char EveProgressBar1_Caption[];
```

```
extern const code far char BoxRoundsmpScn1_Caption[];
extern const code far char EveTextSampScnLabel1_Caption[];
extern const code far char EveTextsampscnlabel2_Caption[];
extern const code far char EveTextsampscnlabel4_Caption[];
extern const code far char EveTextsampscnlabel3_Caption[];
extern const code far char EveTextsampscnlabel5_Caption[];
extern const code far char EveTextsampscnlabel6_Caption[];
extern const code far char BoxRoundSmpscn2_Caption[];
extern char EveTextEncoderSamp_Caption[];
extern char EveTextLaserSmp_Caption[];
extern char EveTextPitchSmp_Caption[];
extern char EveTextGPSSmp_Caption[];
extern char EveTextSDSmp_Caption[];
extern char EveTextRollSmp_Caption[];
extern char EveTextAccx_Caption[];
extern char EveTextAccy_Caption[];
extern char EveTextAccz_Caption[];
extern char BoxRound7_Caption[];
extern char EveButtonsmpJogRear_Caption[];
extern char EveButtonsmpJogFront_Caption[];
extern char EveButtonsmpBack_Caption[];
extern char BoxRound8_Caption[];
extern char EveButtonSmpStartSampling_Caption[];
extern char EveButtonSmpStopSampling_Caption[];
extern char EveButtonSmpConfigureMotor_Caption[];
extern char EveButtonSmpGotoProf_Caption[];
extern char EveButtonsmpResetInc_Caption[];
extern const code far char Box1_Caption[];
extern const code far char Box2_Caption[];
extern const code far char EveTextProflabel1_Caption[];
extern const code far char EveTextProflabel2_Caption[];
extern const code far char BoxRound1_Caption[];
extern char EveGaugeProfSpeed_Caption[];
extern char EveGaugeProfCrossSlp_Caption[];
extern const code far char EveTextProflabel5_Caption[];
extern const code far char EveTextProflabel4_Caption[];
extern char EveToggleProfMode_StateOFF_Caption[];
extern char EveToggleProfMode_StateON_Caption[];
extern const code far char EveTextProflabel3_Caption[];
extern char EveButtonProfStartProfiling_Caption[];
extern char EveButtonProfStopProfiling_Caption[];
extern char Line1_Caption[];
extern char Line2_Caption[];
extern char Line3_Caption[];
extern char Line4_Caption[];
extern char Line5_Caption[];
extern char Line6_Caption[];
extern char Line7_Caption[];
extern char Line8_Caption[];
extern char Line9_Caption[];
extern char Line10_Caption[];
extern char Line11_Caption[];
extern char Line12_Caption[];
extern char Line13_Caption[];
extern char Line14_Caption[];
extern char Line15_Caption[];
extern char Line16_Caption[];
extern char Line17_Caption[];
extern char Line18_Caption[];
extern char Line19_Caption[];
extern char Line20_Caption[];
extern char Line21_Caption[];
extern char Line22_Caption[];
extern char Line23_Caption[];
extern char Line24_Caption[];
extern char Line25_Caption[];
extern char Line26_Caption[];
extern char Line27_Caption[];
extern char Line28_Caption[];
extern char Line29_Caption[];
extern char Line30_Caption[];
extern char Line31_Caption[];
extern char Line32_Caption[];
extern char Line33_Caption[];
extern char Line34_Caption[];
extern char Line35_Caption[];
extern char Line36_Caption[];
extern char Line37_Caption[];
extern char Line38_Caption[];
extern char Line39_Caption[];
extern char Line40_Caption[];
extern char Line41_Caption[];
extern char Line42_Caption[];
extern char Line43_Caption[];
extern char Line44_Caption[];
extern char Line45_Caption[];
extern char Line46_Caption[];
extern char Line47_Caption[];
extern char Line48_Caption[];
extern char Line49_Caption[];
extern char Line50_Caption[];
extern char EveText1_Caption[];
extern char EveButtonProfCreateFile_Caption[];
extern const code far char EveText10_Caption[];
extern char EveTextProfFileName_Caption[];
extern const code far char EveText12_Caption[];
extern char BoxProfWaitingForCam_Caption[];
extern char EveTextProfWaitingForCam_Caption[];
extern char EveButtonFNfilename_Caption[];
extern const code far char BoxRound2_Caption[];
extern char EveKeys1_Caption[];
extern char EveKeys2_Caption[];
extern char EveKeys3_Caption[];
extern char EveKeys4_Caption[];
extern const code far char EveText2_Caption[];
extern const code far char BoxRound3_Caption[];
extern char EveButtonFNBackspace_Caption[];
extern char EveButtonFNClear_Caption[];
extern char EveButtonFNConfirm_Caption[];
extern char EveButtonFNGoBack_Caption[];
extern char EveButtonFNShift_Caption[];
extern const code far char BoxRound4_Caption[];
extern const code far char EveText3_Caption[];
extern const code far char EveText4_Caption[];
extern const code far char EveText5_Caption[];
extern const code far char EveText6_Caption[];
extern const code far char EveText7_Caption[];
extern const code far char BoxRound5_Caption[];
extern char EveButtonSUNewRun_Caption[];
extern char EveButtonSUHelp_Caption[];
extern char EveButtonSUAbout_Caption[];
extern char EveTextSUtime_Caption[];
extern char EveTextSUdistance_Caption[];
extern char EveTextSUSpeed_Caption[];
extern char EveTextSUFilename_Caption[];
extern char EveTextSUAverageSpeed_Caption[];
extern char EveText8_Caption[];
extern char BoxRound6_Caption[];
extern char EveText9_Caption[];
extern char EveText11_Caption[];
extern char EveText13_Caption[];
extern char EveText14_Caption[];
extern char EveText15_Caption[];
extern char EveText16_Caption[];
extern char EveText17_Caption[];
extern char EveButtonSUBack_Caption[];

extern TEvent EveButtonSplashInit_OnUpOnClick;
```

```
extern TEvent EveButtonsmpJogRear_OnUpOnPress;
extern TEvent EveButtonsmpJogFront_OnUpOnPress;
extern TEvent EveButtonsmpBack_OnUpOnClick;
extern TEvent EveButtonSmpStartSampling_OnUpOnClick;
extern TEvent EveButtonSmpStopSampling_OnUpOnClick;
extern TEvent EveButtonSmpConfigureMotor_OnUpOnClick;
extern TEvent EveButtonSmpGotoProf_OnUpOnClick;
extern TEvent EveButtonsmpResetInc_OnUpOnClick;
extern TEvent EveToggleProfMode_OnUpOnClick;
extern TEvent EveButtonProfStartProfiling_OnUpOnClick;
extern TEvent EveButtonProfStopProfiling_OnUpOnClick;
extern TEvent EveButtonProfCreateFile_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNBackspace_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNClear_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNConfirm_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNGoBack_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNShift_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonSUNewRun_OnUpOnClick;
extern TEvent EveButtonSUHelp_OnUpOnClick;
extern TEvent EveButtonSUAbout_OnUpOnClick;
extern TEvent EveButtonSUBack_OnUpOnClick;


void DrawScreenO(TScreen *aScreen, char aOptions);
void DrawScreen(TScreen *aScreen);
void DrawBox(TBox *ABox);
void DrawCBox(TCBox *ACBox);
void DrawBoxRound(TBox_Round *ABoxRound);
void DrawCBoxRound(TCBox_Round *ACBoxRound);
void DrawLine(TLine *ALine);
void DrawEveGauge(TEveGauge *AEveGauge);
void DrawEveKeys(TEveKeys *AEveKeys);
void DrawEveProgressBar(TEveProgressBar *AEveProgressBar);
void DrawEveToggle(TEveToggle *AEveToggle);
void DrawEveButton(TEveButton *AEveButton);
void DrawEveText(TEveText *AEveText);
void DrawCEveText(TCEveText *ACEveText);
void DrawCEveNumber(TCEveNumber *ACEveNumber);
void ProcessVTFTStack();
void InitVTFTStack();
#line 24
"C:/Users/ipm4/Desktop/latest_code_01062014/0121/NewDesignF
T800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/NewDesign_main.c"
void main() {
 InitVTFTStack();
 while (1) {
 ProcessVTFTStack();
 }

}
```

NEWDESIGN_DRIVER.CP

```
#line 1
"C:/Users/ipm4/Desktop/latest_code_01062014/0121/NewDesignF
T800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/NewDesign_driver.c"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_objects.h"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
#line 21
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
typedef unsigned char uint8_ft8;
typedef signed char int8_ft8;
typedef unsigned int uint16_ft8;
typedef int int16_ft8;
typedef unsigned long uint32_ft8;
typedef long int32_ft8;
typedef unsigned long long uint64_ft8;
typedef long long int64_ft8;

typedef enum {cmdREAD, cmdWRITE} TCmd;

typedef enum {opNONE, opREAD, opWRITE} TOpr;

typedef enum {umNONE, umDL=1<<0, umCP=1<<1,
umGRAM=1<<2} TUpdateMode;

typedef uint8_ft8 TFT800Alpha;
typedef uint8_ft8 TFT800Handle;

typedef struct
{
 int16_ft8 X;
 int16_ft8 Y;
} TFT800Point;

typedef struct
{
 int16_ft8 Left;
 int16_ft8 Top;
 uint16_ft8 Width;
 uint16_ft8 Height;
} TFT800Rect;

typedef union
{
 struct
 {
 uint8_ft8 B;
 uint8_ft8 G;
 uint8_ft8 R;
 };
 uint32_ft8 RGB;
} TFT800Color;

typedef struct
{
 TFT800Color Color;
 TFT800Alpha Alpha;
 uint8_ft8 Width;
} TFT800Pen;

typedef struct
{
 TFT800Color ColorBg;
 TFT800Color Color;


 TFT800Color ColorTo;
 uint8_ft8 Gradient;
 TFT800Alpha Alpha;
 uint8_ft8 Style;
} TFT800Brush;



typedef struct
{

 uint8_ft8 FontWidth[ 256 ];
 uint32_ft8 FontBitmapFormat;
 uint32_ft8 FontLineStride;
 uint32_ft8 FontWidthInPixels;
 uint32_ft8 FontHeightInPixels;
 uint32_ft8 PointerToFontGraphicsData;
} TFT800FontGraphics;

typedef struct
{
 TFT800FontGraphics GrData;
 uint16_ft8 FirstChar;
 uint16_ft8 LastChar;
} TFT800CharSetDsc;

typedef struct
{
 uint32_ft8 Name;
 uint32_ft8 Source;
 TFT800Color Color;
 TFT800Alpha Alpha;
 TFT800Handle Handle;
 TFT800CharSetDsc CharSetDsc;
} TFT800Font;

typedef struct
{
 uint8_ft8 Format;
 uint16_ft8 LineStride;
 uint16_ft8 Height;
} TFT800BmpLayout;

typedef struct
{
 uint8_ft8 FwFlags;
 uint16_ft8 Width;
 uint16_ft8 Height;
} TFT800BmpSize;

typedef struct
{
 uint32_ft8 Source;
 TFT800BmpLayout Layout;
 TFT800BmpSize Size;
} TFT800BmpConfig;

typedef struct
{
 TFT800BmpConfig Configs[16];
 uint8_ft8 Current;
} TFT800BmpHandle;

typedef struct
```

```
{
 TFT800BmpConfig Config;
 TFT800Color BlendColor;
 TFT800Alpha Alpha;
 TFT800Handle Handle;
} TFT800Bitmap;

typedef struct
{
 uint16_ft8 Hour;
 uint16_ft8 Min;
 uint16_ft8 Sec;
 uint16_ft8 mSec;
} TFT800Time;

typedef struct
{
 uint32_ft8 DrawingOptions;
 uint16_ft8 Val;
 uint16_ft8 Size;
 uint16_ft8 Range;
 uint16_ft8 Minor;
 uint16_ft8 Major;
 TFT800Time Time;
 uint8_ft8 Style;
} TFT800CoProcGraphics;

typedef struct
{
 TFT800Pen Pen;
 TFT800Brush Brush;
 TFT800Font Font;
 TFT800Bitmap Bitmap;

 TFT800Rect ClipRect;
 TFT800Point Cursor;

 uint8_ft8 Stencil;
 uint8_ft8 Tag;

 TFT800CoProcGraphics CPGraphics;
} TFT800Canvas;

typedef struct
{
 TOpr Opr;
 uint32_ft8 RWPtr;
} TFT800IO;


typedef struct
{

 uint32_ft8 Frequency;
 uint32_ft8 OutRenderMode;
 uint32_ft8 RenderReadScanLine;
 uint32_ft8 RenderWriteTrigger;

 uint32_ft8 hCycle;
 uint32_ft8 hOffset;
 uint32_ft8 hSize;
 uint32_ft8 hSync0;
 uint32_ft8 hSync1;

 uint32_ft8 vCycle;
 uint32_ft8 vOffset;
 uint32_ft8 vSize;
 uint32_ft8 vSync0;
 uint32_ft8 vSync1;
 uint32_ft8 Rotate180;
 uint32_ft8 OutBits;
 uint32_ft8 OutDither;
 uint32_ft8 OutSwizzle;
 uint32_ft8 OutCSpread;
 uint32_ft8 PClockPolarity;
 uint32_ft8 PClock;
} TFT800Display;

typedef struct
{
 uint32_ft8 TouchMode;
 uint32_ft8 TouchADCMode;
 uint32_ft8 TouchCharge;
 uint32_ft8 TouchSettle;
 uint32_ft8 TouchOversample;
 uint32_ft8 TouchRZThreshold;
} TFT800Touch;

typedef struct
{
 uint32_ft8 TransformA;
 uint32_ft8 TransformB;
 uint32_ft8 TransformC;
 uint32_ft8 TransformD;
 uint32_ft8 TransformE;
 uint32_ft8 TransformF;
} TFT800TouchTransform;

typedef struct
{
 uint8_ft8 Enable;
 uint8_ft8 Mask;
 uint8_ft8 Flags;
} TFT800Interrupt;

typedef struct
{
 uint8_ft8 Effect;
 uint8_ft8 Pitch;
 uint8_ft8 Volume;
 uint8_ft8 Play;
} TFT800Sound;

typedef struct
{
 uint32_ft8 StartAddress;
 uint32_ft8 Length;
 uint16_ft8 Frequency;
 uint8_ft8 Format;
 uint8_ft8 Loop;
 uint8_ft8 Volume;
 uint8_ft8 Play;
} TFT800Audio;

typedef struct
{

 uint16_ft8 Freq;
 uint8_ft8 Duty;
} TFT800PWM;

typedef struct
{
 uint8_ft8 GPIODIR;
 uint8_ft8 GPIO;
} TFT800GPIO;

typedef struct
{
```

```
 TFT800PWM *pPWMCfg;
 TFT800GPIO *pGPIOCfg;
 TFT800Audio *pAudioCfg;
 TFT800Sound *pSoundCfg;
 TFT800Touch *pTouchCfg;
 TFT800Display *pDisplayCfg;
 TFT800Interrupt *pInterruptCfg;
 TFT800TouchTransform *pTTransformCfg;
} TFT800Config;



typedef struct
{
 struct
 {




 TFT800Color Color;
 TFT800Alpha Alpha;

 uint8_ft8 Tag;

 TFT800Color ClearColor;
 TFT800Alpha ClearAlpha;
 uint8_ft8 ClearStencil;
 uint8_ft8 ClearTag;

 uint16_ft8 LineWidth;
 uint16_ft8 PointSize;

 uint16_ft8 ScissorLeft;
 uint16_ft8 ScissorTop;
 uint16_ft8 ScissorWidth;
 uint16_ft8 ScissorHeight;

 uint8_ft8 BmpHandle;
 uint8_ft8 Cell;

 struct
 {
 Color : 1;
 Alpha : 1;
 Tag : 1;
 ClearColor : 1;
 ClearAlpha : 1;
 ClearStencil : 1;
 ClearTag : 1;
 LineWidth : 1;
 PointSize : 1;
 ScissorPos : 1;
 ScissorSize : 1;
 BmpHandle : 1;
 Cell : 1;

 Unused : 32-13;

 } UpdateFlags;


 } Context;

 TFT800BmpConfig BmpHandleCfg[16];

 int8_ft8 CurrGrPrim;
} TFT800Graphics;

typedef struct
{
 TFT800IO IO;

 TFT800Sound Sound;

 TFT800Audio Audio;

 TFT800Graphics Graphics;

 struct
 { uint16_ft8 Width;
 uint16_ft8 Height;
 } Display;

 uint8_ft8 UpdateMode;

} TFT800Controller;
#line 6
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_objects.h"
typedef enum {taNone, taLeft, taCenter, taRight, taCenterX,
taCenterY, taRightX} TTextAlign;

typedef struct Screen TScreen;

typedef unsigned long TPointer;

typedef struct tagObjInfo {
 TPointer Obj;
 char Type;
 char Order;
 char Flags;

 char HitTag;
 int HitX;
 int HitY;
} TObjInfo;

typedef struct tagTouchStat {
 char Pressed;

 char Tag;
 int X;
 int Y;

 TObjInfo ActObjInfo;
} TTouchStat;

typedef void (*TDrawHandler)(TPointer aObj);

typedef void (*TEvtAction)();

typedef struct tagEvtSound {
 char SndAct;
 char Effect;
 char Pitch;
 char Volume;
} TEvtSound;

typedef const far struct tagCEvent {
 TEvtAction Action;
 TEvtSound Sound;
} TCEvent;

typedef struct tagEvent {
 TEvtAction Action;
 TEvtSound Sound;
} TEvent;
```

```
typedef const far struct tagCRect {
 int Left;
 int Top;
 int Width;
 int Height;
} TCRect;

typedef struct tagRect {
 int Left;
 int Top;
 int Width;
 int Height;
} TRect;

typedef struct tagBox {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char Gradient;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TBox;

typedef far const code struct tagCBox {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char Gradient;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCBox;

typedef struct tagBox_Round {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Corner_Radius;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TBox_Round;

typedef far const code struct tagCBox_Round {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Corner_Radius;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCBox_Round;

typedef struct tagLine {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int First_Point_X;
 int First_Point_Y;
 int Second_Point_X;
 int Second_Point_Y;
 char Pen_Width;
 unsigned int Pen_Color;
} TLine;

typedef struct tagEveGauge {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Radius;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char Major;
 char Minor;
 unsigned int Value;
 unsigned int Range;
 char Flat;
 char NoBackground;
 char NoPointer;
```

```
 char TicksVisible;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveGauge;

typedef struct tagEveKeys {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char *Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Flat;
 char AutoSize;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveKeys;

typedef struct tagEveProgressBar {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Background_Color;
 unsigned int Color;
 unsigned int Value;
 unsigned int Range;
 char Flat;
} TEveProgressBar;

typedef struct tagEveToggle {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Pen_Width;
 unsigned int Pen_Color;
 unsigned int Background_Color;
 unsigned int Color;
 unsigned int Press_Color;
 char *StateOFF_Caption;
 char *StateON_Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 unsigned int State;
 char Flat;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveToggle;

typedef struct tagEveButton {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 unsigned int Color;
 unsigned int Press_Color;
 unsigned int ColorTo;
 unsigned int Press_ColorTo;
 char *Caption;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Flat;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveButton;

typedef struct tagEveText {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char *Caption;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Active;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnClick;
 TEvent *OnPress;
} TEveText;

typedef far const code struct tagCEveText {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
```

```
 int Top;
 int Width;
 int Height;
 far const code char *Caption;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCEveText;

typedef far const code struct tagCEveNumber {
 TScreen *OwnerScreen;
 char Order;
 char Visible;
 char Opacity;
 char Tag;
 int Left;
 int Top;
 int Width;
 int Height;
 char Text_Length;
 TTextAlign TextAlign;
 far const code char *FontName;
 unsigned int Font_Color;
 char FontHandle;
 long Source;
 signed long Value;
 unsigned char Signed;
 char Active;
 TCEvent *OnUp;
 TCEvent *OnDown;
 TCEvent *OnClick;
 TCEvent *OnPress;
} TCEveNumber;

struct Screen {
 unsigned int Color;
 unsigned int Width;
 unsigned int Height;
 unsigned short ObjectsCount;
 unsigned short BoxesCount;
 TBox *far const code *Boxes;
 unsigned short CBoxesCount;
 TCBox *far const code *CBoxes;
 unsigned short Boxes_RoundCount;
 TBox_Round *far const code *Boxes_Round;
 unsigned short CBoxes_RoundCount;
 TCBox_Round *far const code *CBoxes_Round;
 unsigned short LinesCount;
 TLine *far const code *Lines;
 unsigned short EveGaugesCount;
 TEveGauge *far const code *EveGauges;
 unsigned short EveKeysCount;
 TEveKeys *far const code *EveKeys;
 unsigned short EveProgressBarsCount;
 TEveProgressBar *far const code *EveProgressBars;
 unsigned short EveTogglesCount;
 TEveToggle *far const code *EveToggles;
 unsigned short EveButtonsCount;
 TEveButton *far const code *EveButtons;
 unsigned short EveTextsCount;
 TEveText *far const code *EveTexts;
 unsigned short CEveTextsCount;
 TCEveText *far const code *CEveTexts;
 unsigned short CEveNumbersCount;
 TCEveNumber *far const code *CEveNumbers;
 unsigned long DynResStart;
 unsigned short Active;
 unsigned short SniffObjectEvents;
 TEvent *OnUp;
 TEvent *OnDown;
 TEvent *OnTagChange;
 TEvent *OnPress;
};

extern const VTFT_OT_BOX;
extern const VTFT_OT_CBOX;
extern const VTFT_OT_BOXROUND;
extern const VTFT_OT_CBOXROUND;
extern const VTFT_OT_LINE;
extern const VTFT_OT_EVEGAUGE;
extern const VTFT_OT_EVEKEYS;
extern const VTFT_OT_EVEPROGRESSBAR;
extern const VTFT_OT_EVETOGGLE;
extern const VTFT_OT_EVEBUTTON;
extern const VTFT_OT_EVETEXT;
extern const VTFT_OT_CEVETEXT;
extern const VTFT_OT_CEVENUMBER;

extern const VTFT_EVT_UP;
extern const VTFT_EVT_DOWN;
extern const VTFT_EVT_CLICK;
extern const VTFT_EVT_PRESS;

extern const VTFT_SNDACT_NONE;
extern const VTFT_SNDACT_PLAY;
extern const VTFT_SNDACT_STOP;

extern const VTFT_LOAD_RES_NONE;
extern const VTFT_LOAD_RES_STATIC;
extern const VTFT_LOAD_RES_DYNAMIC;
extern const VTFT_LOAD_RES_ALL;

extern const VTFT_DISPLAY_EFF_NONE;
extern const VTFT_DISPLAY_EFF_LIGHTS_FADE;
extern const VTFT_DISPLAY_EFF_LIGHTS_OFF;

extern const VTFT_INT_REPAINT_ON_DOWN;
extern const VTFT_INT_REPAINT_ON_UP;
extern const VTFT_INT_BRING_TO_FRONT;
extern const VTFT_INT_INTRINSIC_CLICK_EFF;

extern const TPointer DrawHandlerTable[44];
```

```
extern const TFT800PWM VTFT_FT800_CONFIG_PWM;
extern const TFT800GPIO VTFT_FT800_CONFIG_GPIO;
extern const TFT800Sound VTFT_FT800_CONFIG_SOUND;
extern const TFT800Audio VTFT_FT800_CONFIG_AUDIO;
extern const TFT800Display VTFT_FT800_CONFIG_DISPLAY;
extern const TFT800Interrupt
VTFT_FT800_CONFIG_INTERRUPT;
extern const TFT800Touch VTFT_FT800_CONFIG_TOUCH;
extern const TFT800TouchTransform
VTFT_FT800_CONFIG_TOUCHTRANSFORM;

extern TTouchStat TouchS;

extern TScreen SplashScreen;

extern TCEveText EveTextSplashScreenName;
extern TEveButton EveButtonSplashInit;
extern TEvent EveButtonSplashInit_OnClick;
extern TEveText EveTextSplashScreenStatus;
extern TEveProgressBar EveProgressBar1;

extern TEveProgressBar *far const code
SplashScreen_EveProgressBars[1];
extern TEveButton *far const code SplashScreen_EveButtons[1];
extern TEveText *far const code SplashScreen_EveTexts[1];
extern TCEveText *far const code SplashScreen_CEveTexts[1];

extern TScreen SamplingScreen;

extern TCBox_Round BoxRoundsmpScn1;
extern TCEveText EveTextSampScnLabel1;
extern TCEveText EveTextsampscnlabel2;
extern TCEveText EveTextsampscnlabel4;
extern TCEveText EveTextsampscnlabel3;
extern TCEveText EveTextsampscnlabel5;
extern TCEveText EveTextsampscnlabel6;
extern TCBox_Round BoxRoundSmpscn2;
extern TEveText EveTextEncoderSamp;
extern TEveText EveTextLaserSmp;
extern TEveText EveTextPitchSmp;
extern TEveText EveTextGPSSmp;
extern TEveText EveTextSDSmp;
extern TEveText EveTextRollSmp;
extern TEveText EveTextAccx;
extern TEveText EveTextAccy;
extern TEveText EveTextAccz;
extern TBox_Round BoxRound7;
extern TEveButton EveButtonsmpJogRear;
extern TEvent EveButtonsmpJogRear_OnPress;
extern TEveButton EveButtonsmpJogFront;
extern TEvent EveButtonsmpJogFront_OnPress;
extern TEveButton EveButtonsmpBack;
extern TEvent EveButtonsmpBack_OnClick;
extern TBox_Round BoxRound8;
extern TEveButton EveButtonSmpStartSampling;
extern TEvent EveButtonSmpStartSampling_OnClick;
extern TEveButton EveButtonSmpStopSampling;
extern TEvent EveButtonSmpStopSampling_OnClick;
extern TEveButton EveButtonSmpConfigureMotor;
extern TEvent EveButtonSmpConfigureMotor_OnClick;
extern TEveButton EveButtonSmpGotoProf;
extern TEvent EveButtonSmpGotoProf_OnClick;
extern TEveButton EveButtonsmpResetInc;
extern TEvent EveButtonsmpResetInc_OnClick;

extern TBox_Round *far const code
SamplingScreen_Boxes_Round[2];
extern TCBox_Round *far const code
SamplingScreen_CBoxes_Round[2];
extern TEveButton *far const code
SamplingScreen_EveButtons[8];
extern TEveText *far const code SamplingScreen_EveTexts[9];
extern TCEveText *far const code SamplingScreen_CEveTexts[6];

extern TScreen ProfilingScreen;

extern TCBox Box1;
extern TCBox Box2;
extern TCEveText EveTextProflabel1;
extern TCEveText EveTextProflabel2;
extern TCEveNumber EveNumber1;
extern TCEveNumber EveNumber2;
extern TCEveNumber EveNumber3;
extern TCEveNumber EveNumber4;
extern TCEveNumber EveNumber5;
extern TCEveNumber EveNumber6;
extern TCBox_Round BoxRound1;
extern TEveGauge EveGaugeProfSpeed;
extern TEveGauge EveGaugeProfCrossSlp;
extern TCEveText EveTextProflabel5;
extern TCEveText EveTextProflabel4;
extern TEveToggle EveToggleProfMode;
extern TEvent EveToggleProfMode_OnClick;
extern TCEveText EveTextProflabel3;
extern TEveButton EveButtonProfStartProfiling;
extern TEvent EveButtonProfStartProfiling_OnClick;
extern TEveButton EveButtonProfStopProfiling;
extern TEvent EveButtonProfStopProfiling_OnClick;
extern TLine Line1;
extern TLine Line2;
extern TLine Line3;
extern TLine Line4;
extern TLine Line5;
extern TLine Line6;
extern TLine Line7;
extern TLine Line8;
extern TLine Line9;
extern TLine Line10;
extern TLine Line11;
extern TLine Line12;
extern TLine Line13;
extern TLine Line14;
extern TLine Line15;
extern TLine Line16;
extern TLine Line17;
extern TLine Line18;
extern TLine Line19;
extern TLine Line20;
extern TLine Line21;
extern TLine Line22;
extern TLine Line23;
extern TLine Line24;
extern TLine Line25;
extern TLine Line26;
extern TLine Line27;
extern TLine Line28;
extern TLine Line29;
extern TLine Line30;
extern TLine Line31;
extern TLine Line32;
extern TLine Line33;
extern TLine Line34;
extern TLine Line35;
extern TLine Line36;
extern TLine Line37;
extern TLine Line38;
extern TLine Line39;
```

```
extern TLine Line40;
extern TLine Line41;
extern TLine Line42;
extern TLine Line43;
extern TLine Line44;
extern TLine Line45;
extern TLine Line46;
extern TLine Line47;
extern TLine Line48;
extern TLine Line49;
extern TLine Line50;
extern TEveText EveText1;
extern TEveButton EveButtonProfCreateFile;
extern TEvent EveButtonProfCreateFile_OnClick;
extern TCEveText EveText10;
extern TEveText EveTextProfFileName;
extern TCEveText EveText12;
extern TBox BoxProfWaitingForCam;
extern TEveText EveTextProfWaitingForCam;

extern TBox *far const code ProfilingScreen_Boxes[1];
extern TCBox *far const code ProfilingScreen_CBoxes[2];
extern TCBox_Round *far const code
ProfilingScreen_CBoxes_Round[1];
extern TLine *far const code ProfilingScreen_Lines[50];
extern TEveGauge *far const code ProfilingScreen_EveGauges[2];
extern TEveToggle *far const code
ProfilingScreen_EveToggles[1];
extern TEveButton *far const code
ProfilingScreen_EveButtons[3];
extern TEveText *far const code ProfilingScreen_EveTexts[3];
extern TCEveText *far const code ProfilingScreen_CEveTexts[7];
extern TCEveNumber *far const code
ProfilingScreen_CEveNumbers[6];

extern TScreen SettingScreen;
extern TEvent SettingScreen_OnTagChange;

extern TEveButton EveButtonFNfilename;
extern TCBox_Round BoxRound2;
extern TEveKeys EveKeys1;
extern TEveKeys EveKeys2;
extern TEveKeys EveKeys3;
extern TEveKeys EveKeys4;
extern TCEveText EveText2;
extern TCBox_Round BoxRound3;
extern TEveButton EveButtonFNBackspace;
extern TEvent EveButtonFNBackspace_OnClick;
extern TEveButton EveButtonFNClear;
extern TEvent EveButtonFNClear_OnClick;
extern TEveButton EveButtonFNConfirm;
extern TEvent EveButtonFNConfirm_OnClick;
extern TEveButton EveButtonFNGoBack;
extern TEvent EveButtonFNGoBack_OnClick;
extern TEveButton EveButtonFNShift;
extern TEvent EveButtonFNShift_OnClick;

extern TCBox_Round *far const code
SettingScreen_CBoxes_Round[2];
extern TEveKeys *far const code SettingScreen_EveKeys[4];
extern TEveButton *far const code SettingScreen_EveButtons[6];
extern TCEveText *far const code SettingScreen_CEveTexts[1];

extern TScreen SummaryScreen;

extern TCBox_Round BoxRound4;
extern TCEveText EveText3;
extern TCEveText EveText4;
extern TCEveText EveText5;
extern TCEveText EveText6;
extern TCEveText EveText7;
extern TCBox_Round BoxRound5;
extern TEveButton EveButtonSUNewRun;
extern TEvent EveButtonSUNewRun_OnClick;
extern TEveButton EveButtonSUHelp;
extern TEvent EveButtonSUHelp_OnClick;
extern TEveButton EveButtonSUAbout;
extern TEvent EveButtonSUAbout_OnClick;
extern TEveText EveTextSUtime;
extern TEveText EveTextSUdistance;
extern TEveText EveTextSUSpeed;
extern TEveText EveTextSUFilename;
extern TEveText EveTextSUAverageSpeed;
extern TEveText EveText8;
extern TBox_Round BoxRound6;
extern TEveText EveText9;
extern TEveText EveText11;
extern TEveText EveText13;
extern TEveText EveText14;
extern TEveText EveText15;
extern TEveText EveText16;
extern TEveText EveText17;
extern TEveButton EveButtonSUBack;
extern TEvent EveButtonSUBack_OnClick;

extern TBox_Round *far const code
SummaryScreen_Boxes_Round[1];
extern TCBox_Round *far const code
SummaryScreen_CBoxes_Round[2];
extern TEveButton *far const code
SummaryScreen_EveButtons[4];
extern TEveText *far const code SummaryScreen_EveTexts[13];
extern TCEveText *far const code
SummaryScreen_CEveTexts[5];

extern TScreen *CurrentScreen;

void EveButtonFNBackspaceOnClick();
void EveButtonFNClearOnClick();
void EveButtonFNConfirmOnClick();
void EveButtonFNGoBackOnClick();
void EveButtonFNShiftOnClick();
void EveButtonProfCreateFileOnClick();
void EveButtonProfStartProfilingOnClick();
void EveButtonProfStopProfilingOnClick();
void EveButtonsmpBackOnClick();
void EveButtonSmpConfigureMotorOnClick();
void EveButtonSmpGotoProfOnClick();
void EveButtonsmpJogFrontOnPress();
void EveButtonsmpJogRearOnPress();
void EveButtonsmpResetIncOnClick();
void EveButtonSmpStartSamplingOnClick();
void EveButtonSmpStopSamplingOnClick();
void EveButtonSplashInitOnClick();
void EveButtonSUAboutOnClick();
void EveButtonSUBackOnClick();
void EveButtonSUHelpOnClick();
void EveButtonSUNewRunOnClick();
void EveToggleProfModeOnClick();
void SettingScreenOnTagChange();

extern const code far char EveTextSplashScreenName_Caption[];
extern char EveButtonSplashInit_Caption[];
extern char EveTextSplashScreenStatus_Caption[];
extern char EveProgressBar1_Caption[];
```

```
extern const code far char BoxRoundsmpScn1_Caption[];
extern const code far char EveTextSampScnLabel1_Caption[];
extern const code far char EveTextsampscnlabel2_Caption[];
extern const code far char EveTextsampscnlabel4_Caption[];
extern const code far char EveTextsampscnlabel3_Caption[];
extern const code far char EveTextsampscnlabel5_Caption[];
extern const code far char EveTextsampscnlabel6_Caption[];
extern const code far char BoxRoundSmpscn2_Caption[];
extern char EveTextEncoderSamp_Caption[];
extern char EveTextLaserSmp_Caption[];
extern char EveTextPitchSmp_Caption[];
extern char EveTextGPSSmp_Caption[];
extern char EveTextSDSmp_Caption[];
extern char EveTextRollSmp_Caption[];
extern char EveTextAccx_Caption[];
extern char EveTextAccy_Caption[];
extern char EveTextAccz_Caption[];
extern char BoxRound7_Caption[];
extern char EveButtonsmpJogRear_Caption[];
extern char EveButtonsmpJogFront_Caption[];
extern char EveButtonsmpBack_Caption[];
extern char BoxRound8_Caption[];
extern char EveButtonSmpStartSampling_Caption[];
extern char EveButtonSmpStopSampling_Caption[];
extern char EveButtonSmpConfigureMotor_Caption[];
extern char EveButtonSmpGotoProf_Caption[];
extern char EveButtonsmpResetInc_Caption[];
extern const code far char Box1_Caption[];
extern const code far char Box2_Caption[];
extern const code far char EveTextProflabel1_Caption[];
extern const code far char EveTextProflabel2_Caption[];
extern const code far char BoxRound1_Caption[];
extern char EveGaugeProfSpeed_Caption[];
extern char EveGaugeProfCrossSlp_Caption[];
extern const code far char EveTextProflabel5_Caption[];
extern const code far char EveTextProflabel4_Caption[];
extern char EveToggleProfMode_StateOFF_Caption[];
extern char EveToggleProfMode_StateON_Caption[];
extern const code far char EveTextProflabel3_Caption[];
extern char EveButtonProfStartProfiling_Caption[];
extern char EveButtonProfStopProfiling_Caption[];
extern char Line1_Caption[];
extern char Line2_Caption[];
extern char Line3_Caption[];
extern char Line4_Caption[];
extern char Line5_Caption[];
extern char Line6_Caption[];
extern char Line7_Caption[];
extern char Line8_Caption[];
extern char Line9_Caption[];
extern char Line10_Caption[];
extern char Line11_Caption[];
extern char Line12_Caption[];
extern char Line13_Caption[];
extern char Line14_Caption[];
extern char Line15_Caption[];
extern char Line16_Caption[];
extern char Line17_Caption[];
extern char Line18_Caption[];
extern char Line19_Caption[];
extern char Line20_Caption[];
extern char Line21_Caption[];
extern char Line22_Caption[];
extern char Line23_Caption[];
extern char Line24_Caption[];
extern char Line25_Caption[];
extern char Line26_Caption[];
extern char Line27_Caption[];
extern char Line28_Caption[];
extern char Line29_Caption[];
extern char Line30_Caption[];
extern char Line31_Caption[];
extern char Line32_Caption[];
extern char Line33_Caption[];
extern char Line34_Caption[];
extern char Line35_Caption[];
extern char Line36_Caption[];
extern char Line37_Caption[];
extern char Line38_Caption[];
extern char Line39_Caption[];
extern char Line40_Caption[];
extern char Line41_Caption[];
extern char Line42_Caption[];
extern char Line43_Caption[];
extern char Line44_Caption[];
extern char Line45_Caption[];
extern char Line46_Caption[];
extern char Line47_Caption[];
extern char Line48_Caption[];
extern char Line49_Caption[];
extern char Line50_Caption[];
extern char EveText1_Caption[];
extern char EveButtonProfCreateFile_Caption[];
extern const code far char EveText10_Caption[];
extern char EveTextProfFileName_Caption[];
extern const code far char EveText12_Caption[];
extern char BoxProfWaitingForCam_Caption[];
extern char EveTextProfWaitingForCam_Caption[];
extern char EveButtonFNfilename_Caption[];
extern const code far char BoxRound2_Caption[];
extern char EveKeys1_Caption[];
extern char EveKeys2_Caption[];
extern char EveKeys3_Caption[];
extern char EveKeys4_Caption[];
extern const code far char EveText2_Caption[];
extern const code far char BoxRound3_Caption[];
extern char EveButtonFNBackspace_Caption[];
extern char EveButtonFNClear_Caption[];
extern char EveButtonFNConfirm_Caption[];
extern char EveButtonFNGoBack_Caption[];
extern char EveButtonFNShift_Caption[];
extern const code far char BoxRound4_Caption[];
extern const code far char EveText3_Caption[];
extern const code far char EveText4_Caption[];
extern const code far char EveText5_Caption[];
extern const code far char EveText6_Caption[];
extern const code far char EveText7_Caption[];
extern const code far char BoxRound5_Caption[];
extern char EveButtonSUNewRun_Caption[];
extern char EveButtonSUHelp_Caption[];
extern char EveButtonSUAbout_Caption[];
extern char EveTextSUtime_Caption[];
extern char EveTextSUdistance_Caption[];
extern char EveTextSUSpeed_Caption[];
extern char EveTextSUFilename_Caption[];
extern char EveTextSUAverageSpeed_Caption[];
extern char EveText8_Caption[];
extern char BoxRound6_Caption[];
extern char EveText9_Caption[];
extern char EveText11_Caption[];
extern char EveText13_Caption[];
extern char EveText14_Caption[];
extern char EveText15_Caption[];
extern char EveText16_Caption[];
extern char EveText17_Caption[];
extern char EveButtonSUBack_Caption[];

extern TEvent EveButtonSplashInit_OnUpOnClick;
```

```
extern TEvent EveButtonsmpJogRear_OnUpOnPress;
extern TEvent EveButtonsmpJogFront_OnUpOnPress;
extern TEvent EveButtonsmpBack_OnUpOnClick;
extern TEvent EveButtonSmpStartSampling_OnUpOnClick;
extern TEvent EveButtonSmpStopSampling_OnUpOnClick;
extern TEvent EveButtonSmpConfigureMotor_OnUpOnClick;
extern TEvent EveButtonSmpGotoProf_OnUpOnClick;
extern TEvent EveButtonsmpResetInc_OnUpOnClick;
extern TEvent EveToggleProfMode_OnUpOnClick;
extern TEvent EveButtonProfStartProfiling_OnUpOnClick;
extern TEvent EveButtonProfStopProfiling_OnUpOnClick;
extern TEvent EveButtonProfCreateFile_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNBackspace_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNClear_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNConfirm_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNGoBack_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonFNShift_OnUpOnClick;
extern TEvent SettingScreen_OnUpOnTagChange;
extern TEvent EveButtonSUNewRun_OnUpOnClick;
extern TEvent EveButtonSUHelp_OnUpOnClick;
extern TEvent EveButtonSUAbout_OnUpOnClick;
extern TEvent EveButtonSUBack_OnUpOnClick;


void DrawScreenO(TScreen *aScreen, char aOptions);
void DrawScreen(TScreen *aScreen);
void DrawBox(TBox *ABox);
void DrawCBox(TCBox *ACBox);
void DrawBoxRound(TBox_Round *ABoxRound);
void DrawCBoxRound(TCBox_Round *ACBoxRound);
void DrawLine(TLine *ALine);
void DrawEveGauge(TEveGauge *AEveGauge);
void DrawEveKeys(TEveKeys *AEveKeys);
void DrawEveProgressBar(TEveProgressBar *AEveProgressBar);
void DrawEveToggle(TEveToggle *AEveToggle);
void DrawEveButton(TEveButton *AEveButton);
void DrawEveText(TEveText *AEveText);
void DrawCEveText(TCEveText *ACEveText);
void DrawCEveNumber(TCEveNumber *ACEveNumber);
void ProcessVTFTStack();
void InitVTFTStack();
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/newdesign_resources.h"
#line 1 "c:/users/public/documents/mikroelektronika/mikroc pro
for dspic/include/built_in.h"
#line 1
"c:/users/ipm4/desktop/latest_code_01062014/0121/newdesignft80
0/newdesignft800/newdesign_code/mikroc pro for
dspic/ft800_types.h"
#line 8
"C:/Users/ipm4/Desktop/latest_code_01062014/0121/NewDesignF
T800/NewDesignFT800/NewDesign_Code/mikroC PRO for
DSPIC/NewDesign_driver.c"
sbit FT800_RST at LATB6_bit;
sbit FT800_CS at LATB1_bit;
sbit FT800_RST_Direction at TRISB6_bit;
sbit FT800_CS_Direction at TRISB1_bit;


const VTFT_OT_BOX = 12;
const VTFT_OT_CBOX = 13;
const VTFT_OT_BOXROUND = 14;
const VTFT_OT_CBOXROUND = 15;
const VTFT_OT_LINE = 16;
const VTFT_OT_EVEGAUGE = 25;
const VTFT_OT_EVEKEYS = 27;
const VTFT_OT_EVEPROGRESSBAR = 29;
const VTFT_OT_EVETOGGLE = 31;
const VTFT_OT_EVEBUTTON = 36;
const VTFT_OT_EVETEXT = 40;
const VTFT_OT_CEVETEXT = 41;
const VTFT_OT_CEVENUMBER = 43;


const VTFT_EVT_UP = 0;
const VTFT_EVT_DOWN = 1;
const VTFT_EVT_CLICK = 2;
const VTFT_EVT_PRESS = 3;


const VTFT_SNDACT_NONE = 0;
const VTFT_SNDACT_PLAY = 1;
const VTFT_SNDACT_STOP = 2;


const VTFT_LOAD_RES_NONE = 0x00;
const VTFT_LOAD_RES_STATIC = 0x01;
const VTFT_LOAD_RES_DYNAMIC = 0x02;
const VTFT_LOAD_RES_ALL = 0x03;


const VTFT_DISPLAY_EFF_NONE = 0x00;
const VTFT_DISPLAY_EFF_LIGHTS_FADE = 0x04;
const VTFT_DISPLAY_EFF_LIGHTS_OFF = 0x08;


const VTFT_INT_REPAINT_ON_DOWN = 1 << 0;
const VTFT_INT_REPAINT_ON_UP = 1 << 1;
const VTFT_INT_BRING_TO_FRONT = 1 << 2;
const VTFT_INT_INTRINSIC_CLICK_EFF = 1 << 3;


const TPointer DrawHandlerTable[44] = {
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
```

```
 0,
 0,
 0,
 &DrawBox,
 &DrawCBox,
 &DrawBoxRound,
 &DrawCBoxRound,
 &DrawLine,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 &DrawEveGauge,
 0,
 &DrawEveKeys,
 0,
 &DrawEveProgressBar,
 0,
 &DrawEveToggle,
 0,
 0,
 0,
 0,
 &DrawEveButton,
 0,
 0,
 0,
 &DrawEveText,
 &DrawCEveText,
 0,
 &DrawCEveNumber
};



const TFT800Display VTFT_FT800_CONFIG_DISPLAY =
{
 48000000,
 0,
 0,
 0,
 525,
 43,
 480,
 0,
 41,
 286,
 12,
 272,
 0,
 10,
 0,
 0x01B6,
 0,
 0x0000,
 0,
 1,
 5,
};

const TFT800Touch VTFT_FT800_CONFIG_TOUCH =
{
 3,
 1,
 6000,
 3,
 7,
 2000,
};

const TFT800TouchTransform
VTFT_FT800_CONFIG_TOUCHTRANSFORM =
{
 0x0000808D,
 0x0000003C,
 0xFFF11B12,
 0xFFFFFE39,
 0x00004C9B,
 0xFFF37878,
};

const TFT800GPIO VTFT_FT800_CONFIG_GPIO =
{
 0xFC,
 0xFF,
};

const TFT800PWM VTFT_FT800_CONFIG_PWM =
{
 250,
 66,
};

const TFT800Interrupt VTFT_FT800_CONFIG_INTERRUPT =
{
 0,
 255,
 0,
};

const TFT800Sound VTFT_FT800_CONFIG_SOUND =
{
 0,
 0,
 0,
 0,
};

const TFT800Audio VTFT_FT800_CONFIG_AUDIO =
{
 0,
 0,
 8000,
 0,
 0,
 0,
 0,
};



TTouchStat TouchS = {0};



TScreen *CurrentScreen = 0;


TScreen SplashScreen;

TCEveText EveTextSplashScreenName = {
 &SplashScreen,
 0,
 1,
 255,
 255,
```

```
 187,
 143,
 85,
 17,
 EveTextSplashScreenName_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextSplashScreenName_Caption[14] =
"PathMet V 2.0";
TEveButton EveButtonSplashInit;
TEvent EveButtonSplashInit_OnClick;
char EveButtonSplashInit_Caption[11] = "Initialize";
TEveText EveTextSplashScreenStatus;
char EveTextSplashScreenStatus_Caption[16] = "Starting up....";
TEveProgressBar EveProgressBar1;

TEveProgressBar *far const code
SplashScreen_EveProgressBars[1] = {
 &EveProgressBar1
};

TEveButton *far const code SplashScreen_EveButtons[1] = {
 &EveButtonSplashInit
};

TEveText *far const code SplashScreen_EveTexts[1] = {
 &EveTextSplashScreenStatus
};

TCEveText *far const code SplashScreen_CEveTexts[1] = {
 &EveTextSplashScreenName
};


TScreen SamplingScreen;

TCBox_Round BoxRoundsmpScn1 = {
 &SamplingScreen,
 0,
 1,
 255,
 255,
 6,
 4,
 170,
 174,
 0,
 0x22B3,
 0x19CD,
 0xE71C,
 5,
 0,
 0,
 0,
 0,
 0
};
TCEveText EveTextSampScnLabel1 = {
 &SamplingScreen,
 1,
 1,
 255,
 255,
 29,
 8,
 60,
 21,
 EveTextSampScnLabel1_Caption,
 taNone,
 28,
 0xFFFF,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextSampScnLabel1_Caption[8] =
"Encoder";
TCEveText EveTextsampscnlabel2 = {
 &SamplingScreen,
 2,
 1,
 255,
 255,
 29,
 33,
 84,
 21,
 EveTextsampscnlabel2_Caption,
 taNone,
 28,
 0xFFFF,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextsampscnlabel2_Caption[12] =
"Laser(inch)";
TCEveText EveTextsampscnlabel4 = {
 &SamplingScreen,
 3,
 1,
 255,
 255,
 29,
 87,
 79,
 21,
 EveTextsampscnlabel4_Caption,
 taNone,
 28,
 0xFFFF,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextsampscnlabel4_Caption[10] =
"AccXYZ(g)";
TCEveText EveTextsampscnlabel3 = {
 &SamplingScreen,
```

```
 4,
 1,
 255,
 255,
 29,
 61,
 110,
 21,
 EveTextsampscnlabel3_Caption,
 taNone,
 28,
 0xFFFF,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextsampscnlabel3_Caption[16] =
"Pich/Roll (deg)";
TCEveText EveTextsampscnlabel5 = {
 &SamplingScreen,
 5,
 1,
 255,
 255,
 28,
 113,
 32,
 21,
 EveTextsampscnlabel5_Caption,
 taNone,
 28,
 0xFFFF,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextsampscnlabel5_Caption[4] = "GPS";
TCEveText EveTextsampscnlabel6 = {
 &SamplingScreen,
 6,
 1,
 255,
 255,
 28,
 137,
 67,
 21,
 EveTextsampscnlabel6_Caption,
 taNone,
 28,
 0xFFFF,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextsampscnlabel6_Caption[9] = "SD
cards";
TCBox_Round BoxRoundSmpscn2 = {
 &SamplingScreen,
 7,
 1,
 255,
 255,
 179,
 5,
 295,
 172,
 0,
 0x0000,
 0xC618,
 0xE71C,
 5,
 0,
 0,
 0,
 0,
 0
};
TEveText EveTextEncoderSamp;
char EveTextEncoderSamp_Caption[14] = "Waiting";
TEveText EveTextLaserSmp;
char EveTextLaserSmp_Caption[19] = "Waiting";
TEveText EveTextPitchSmp;
char EveTextPitchSmp_Caption[16] = "Waiting";
TEveText EveTextGPSSmp;
char EveTextGPSSmp_Caption[41] = "Not Ready";
TEveText EveTextSDSmp;
char EveTextSDSmp_Caption[8] = "Waiting";
TEveText EveTextRollSmp;
char EveTextRollSmp_Caption[16] = "Waiting";
TEveText EveTextAccx;
char EveTextAccx_Caption[16] = "Waiting";
TEveText EveTextAccy;
char EveTextAccy_Caption[16] = "Waiting";
TEveText EveTextAccz;
char EveTextAccz_Caption[16] = "Waiting";
TBox_Round BoxRound7;
TEveButton EveButtonsmpJogRear;
TEvent EveButtonsmpJogRear_OnPress;
char EveButtonsmpJogRear_Caption[16] = "Jog to the rear";
TEveButton EveButtonsmpJogFront;
TEvent EveButtonsmpJogFront_OnPress;
char EveButtonsmpJogFront_Caption[17] = "Jog to the front";
TEveButton EveButtonsmpBack;
TEvent EveButtonsmpBack_OnClick;
char EveButtonsmpBack_Caption[5] = "back";
TBox_Round BoxRound8;
TEveButton EveButtonSmpStartSampling;
TEvent EveButtonSmpStartSampling_OnClick;
char EveButtonSmpStartSampling_Caption[15] = "Start
sampling";
TEveButton EveButtonSmpStopSampling;
TEvent EveButtonSmpStopSampling_OnClick;
char EveButtonSmpStopSampling_Caption[14] = "Stop sampling";
TEveButton EveButtonSmpConfigureMotor;
TEvent EveButtonSmpConfigureMotor_OnClick;
char EveButtonSmpConfigureMotor_Caption[10] = "Jog Motor";
TEveButton EveButtonSmpGotoProf;
TEvent EveButtonSmpGotoProf_OnClick;
char EveButtonSmpGotoProf_Caption[13] = "Logging Mode";
TEveButton EveButtonsmpResetInc;
TEvent EveButtonsmpResetInc_OnClick;
char EveButtonsmpResetInc_Caption[17] = "Set ground plane";

TBox_Round *far const code SamplingScreen_Boxes_Round[2] =
{
 &BoxRound7,
 &BoxRound8
```

```
};

TCBox_Round *far const code
SamplingScreen_CBoxes_Round[2] = {
 &BoxRoundsmpScn1,
 &BoxRoundSmpscn2
};

TEveButton *far const code SamplingScreen_EveButtons[8] = {
 &EveButtonsmpJogRear,
 &EveButtonsmpJogFront,
 &EveButtonsmpBack,
 &EveButtonSmpStartSampling,
 &EveButtonSmpStopSampling,
 &EveButtonSmpConfigureMotor,
 &EveButtonSmpGotoProf,
 &EveButtonsmpResetInc
};

TEveText *far const code SamplingScreen_EveTexts[9] = {
 &EveTextEncoderSamp,
 &EveTextLaserSmp,
 &EveTextPitchSmp,
 &EveTextGPSSmp,
 &EveTextSDSmp,
 &EveTextRollSmp,
 &EveTextAccx,
 &EveTextAccy,
 &EveTextAccz
};

TCEveText *far const code SamplingScreen_CEveTexts[6] = {
 &EveTextSampScnLabel1,
 &EveTextsampscnlabel2,
 &EveTextsampscnlabel4,
 &EveTextsampscnlabel3,
 &EveTextsampscnlabel5,
 &EveTextsampscnlabel6
};


TScreen ProfilingScreen;

TCBox Box1 = {
 &ProfilingScreen,
 0,
 1,
 255,
 255,
 32,
 24,
 235,
 102,
 1,
 0x0000,
 0xDEFB,
 0xFFFF,
 0xFFFF,
 0xC618,
 _FT800_BRUSH_GR_NONE,
 0,
 0,
 0,
 0,
 0
};
TCBox Box2 = {
 &ProfilingScreen,
 1,
 1,
 255,
 255,
 32,
 152,
 235,
 52,
 1,
 0x0000,
 0xDEFB,
 0xFFFF,
 0xFFFF,
 0xC618,
 _FT800_BRUSH_GR_NONE,
 0,
 0,
 0,
 0,
 0
};
TCEveText EveTextProflabel1 = {
 &ProfilingScreen,
 2,
 1,
 255,
 255,
 7,
 3,
 68,
 17,
 EveTextProflabel1_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextProflabel1_Caption[12] =
"Laser(inch)";
TCEveText EveTextProflabel2 = {
 &ProfilingScreen,
 3,
 1,
 255,
 255,
 8,
 131,
 118,
 17,
 EveTextProflabel2_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextProflabel2_Caption[19] = "Running
Slope(deg)";
TCEveNumber EveNumber1 = {
 &ProfilingScreen,
 4,
```

```
 1,
 255,
 255,
 16,
 111,
 8,
 17,
 0,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 7,
 0,
 1,
 0,
 0,
 0,
 0
};
TCEveNumber EveNumber2 = {
 &ProfilingScreen,
 5,
 1,
 255,
 255,
 16,
 21,
 8,
 17,
 0,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 5,
 0,
 1,
 0,
 0,
 0,
 0
};
TCEveNumber EveNumber3 = {
 &ProfilingScreen,
 6,
 1,
 255,
 255,
 23,
 69,
 6,
 15,
 0,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 6,
 0,
 1,
 0,
 0,
 0,
 0
};
TCEveNumber EveNumber4 = {
 &ProfilingScreen,
 7,
 1,
 255,
 255,
 20,
 149,
 8,
 17,
 0,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 3,
 0,
 1,
 0,
 0,
 0,
 0
};
TCEveNumber EveNumber5 = {
 &ProfilingScreen,
 8,
 1,
 255,
 255,
 14,
 192,
 14,
 17,
 0,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 -3,
 1,
 1,
 0,
 0,
 0,
 0
};
TCEveNumber EveNumber6 = {
 &ProfilingScreen,
 9,
 1,
 255,
 255,
 23,
 172,
 6,
 15,
 0,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 0,
 0,
 1,
 0,
 0,
 0,
 0
```

```
};
TCBox_Round BoxRound1 = {
 &ProfilingScreen,
 10,
 1,
 255,
 255,
 270,
 2,
 208,
 265,
 1,
 0x0000,
 0xAD75,
 0xE71C,
 3,
 0,
 0,
 0,
 0,
 0
};
TEveGauge EveGaugeProfSpeed;
TEveGauge EveGaugeProfCrossSlp;
TCEveText EveTextProflabel5 = {
 &ProfilingScreen,
 13,
 1,
 255,
 255,
 364,
 6,
 88,
 15,
 EveTextProflabel5_Caption,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextProflabel5_Caption[17] = "Cross
Slope(Deg)";
TCEveText EveTextProflabel4 = {
 &ProfilingScreen,
 14,
 1,
 255,
 255,
 275,
 6,
 61,
 15,
 EveTextProflabel4_Caption,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextProflabel4_Caption[11] =
"Speed(M/S)";
TEveToggle EveToggleProfMode;
TEvent EveToggleProfMode_OnClick;
char EveToggleProfMode_StateOFF_Caption[3] = "RL";
char EveToggleProfMode_StateON_Caption[3] = "IW";
TCEveText EveTextProflabel3 = {
 &ProfilingScreen,
 16,
 1,
 255,
 255,
 275,
 133,
 29,
 15,
 EveTextProflabel3_Caption,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveTextProflabel3_Caption[5] = "Mode";
TEveButton EveButtonProfStartProfiling;
TEvent EveButtonProfStartProfiling_OnClick;
char EveButtonProfStartProfiling_Caption[16] = "Start Profiling";
TEveButton EveButtonProfStopProfiling;
TEvent EveButtonProfStopProfiling_OnClick;
char EveButtonProfStopProfiling_Caption[15] = "Stop Profiling";
TLine Line1;
TLine Line2;
TLine Line3;
TLine Line4;
TLine Line5;
TLine Line6;
TLine Line7;
TLine Line8;
TLine Line9;
TLine Line10;
TLine Line11;
TLine Line12;
TLine Line13;
TLine Line14;
TLine Line15;
TLine Line16;
TLine Line17;
TLine Line18;
TLine Line19;
TLine Line20;
TLine Line21;
TLine Line22;
TLine Line23;
TLine Line24;
TLine Line25;
TLine Line26;
TLine Line27;
TLine Line28;
TLine Line29;
TLine Line30;
TLine Line31;
TLine Line32;
TLine Line33;
TLine Line34;
TLine Line35;
```

```
TLine Line36;
TLine Line37;
TLine Line38;
TLine Line39;
TLine Line40;
TLine Line41;
TLine Line42;
TLine Line43;
TLine Line44;
TLine Line45;
TLine Line46;
TLine Line47;
TLine Line48;
TLine Line49;
TLine Line50;
TEveText EveText1;
char EveText1_Caption[19] = "Profile Data File:";
TEveButton EveButtonProfCreateFile;
TEvent EveButtonProfCreateFile_OnClick;
char EveButtonProfCreateFile_Caption[16] = "Create a Folder";
TCEveText EveText10 = {
 &ProfilingScreen,
 71,
 1,
 255,
 255,
 400,
 26,
 38,
 15,
 EveText10_Caption,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveText10_Caption[7] = "-50~50";
TEveText EveTextProfFileName;
char EveTextProfFileName_Caption[10] = "None";
TCEveText EveText12 = {
 &ProfilingScreen,
 73,
 1,
 255,
 255,
 309,
 26,
 20,
 15,
 EveText12_Caption,
 taNone,
 26,
 0x0148,
 26,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveText12_Caption[4] = "0~2";
TBox BoxProfWaitingForCam;
TEveText EveTextProfWaitingForCam;
char EveTextProfWaitingForCam_Caption[32] = "Waiting For
Camera to finish...";

TBox *far const code ProfilingScreen_Boxes[1] = {
 &BoxProfWaitingForCam
};

TCBox *far const code ProfilingScreen_CBoxes[2] = {
 &Box1,
 &Box2
};

TCBox_Round *far const code ProfilingScreen_CBoxes_Round[1]
= {
 &BoxRound1
};

TLine *far const code ProfilingScreen_Lines[50] = {
 &Line1,
 &Line2,
 &Line3,
 &Line4,
 &Line5,
 &Line6,
 &Line7,
 &Line8,
 &Line9,
 &Line10,
 &Line11,
 &Line12,
 &Line13,
 &Line14,
 &Line15,
 &Line16,
 &Line17,
 &Line18,
 &Line19,
 &Line20,
 &Line21,
 &Line22,
 &Line23,
 &Line24,
 &Line25,
 &Line26,
 &Line27,
 &Line28,
 &Line29,
 &Line30,
 &Line31,
 &Line32,
 &Line33,
 &Line34,
 &Line35,
 &Line36,
 &Line37,
 &Line38,
 &Line39,
 &Line40,
 &Line41,
 &Line42,
 &Line43,
 &Line44,
 &Line45,
 &Line46,
 &Line47,
 &Line48,
 &Line49,
 &Line50
};
```

```
TEveGauge *far const code ProfilingScreen_EveGauges[2] = {
 &EveGaugeProfSpeed,
 &EveGaugeProfCrossSlp
};

TEveToggle *far const code ProfilingScreen_EveToggles[1] = {
 &EveToggleProfMode
};

TEveButton *far const code ProfilingScreen_EveButtons[3] = {
 &EveButtonProfStartProfiling,
 &EveButtonProfStopProfiling,
 &EveButtonProfCreateFile
};

TEveText *far const code ProfilingScreen_EveTexts[3] = {
 &EveText1,
 &EveTextProfFileName,
 &EveTextProfWaitingForCam
};

TCEveText *far const code ProfilingScreen_CEveTexts[7] = {
 &EveTextProflabel1,
 &EveTextProflabel2,
 &EveTextProflabel5,
 &EveTextProflabel4,
 &EveTextProflabel3,
 &EveText10,
 &EveText12
};

TCEveNumber *far const code ProfilingScreen_CEveNumbers[6]
= {
 &EveNumber1,
 &EveNumber2,
 &EveNumber3,
 &EveNumber4,
 &EveNumber5,
 &EveNumber6
};


TScreen SettingScreen;
TEvent SettingScreen_OnTagChange;

TEveButton EveButtonFNfilename;
char EveButtonFNfilename_Caption[11] = "";
TCBox_Round BoxRound2 = {
 &SettingScreen,
 1,
 1,
 255,
 255,
 4,
 89,
 474,
 178,
 1,
 0x0000,
 0xC618,
 0xE71C,
 5,
 0,
 0,
 0,
 0,
 0
};
TEveKeys EveKeys1;
char EveKeys1_Caption[11] = "1234567890";
TEveKeys EveKeys2;
char EveKeys2_Caption[12] = "qwertyuiop";
TEveKeys EveKeys3;
char EveKeys3_Caption[11] = "asdfghjkl";
TEveKeys EveKeys4;
char EveKeys4_Caption[10] = "zxcvbnm_";
TCEveText EveText2 = {
 &SettingScreen,
 6,
 1,
 255,
 255,
 8,
 10,
 226,
 21,
 EveText2_Caption,
 taNone,
 28,
 0x0148,
 28,
 -1UL,
 0,
 0,
 0,
 0,
 0
};
far const code char EveText2_Caption[33] = "Enter the name of
the data file:";
TCBox_Round BoxRound3 = {
 &SettingScreen,
 7,
 1,
 255,
 255,
 4,
 32,
 473,
 54,
 1,
 0x0000,
 0xC618,
 0xE71C,
 5,
 0,
 0,
 0,
 0,
 0
};
TEveButton EveButtonFNBackspace;
TEvent EveButtonFNBackspace_OnClick;
char EveButtonFNBackspace_Caption[10] = "Backspace";
TEveButton EveButtonFNClear;
TEvent EveButtonFNClear_OnClick;
char EveButtonFNClear_Caption[10] = "Clear All";
TEveButton EveButtonFNConfirm;
TEvent EveButtonFNConfirm_OnClick;
char EveButtonFNConfirm_Caption[8] = "Confirm";
TEveButton EveButtonFNGoBack;
TEvent EveButtonFNGoBack_OnClick;
char EveButtonFNGoBack_Caption[8] = "Go back";
TEveButton EveButtonFNShift;
TEvent EveButtonFNShift_OnClick;
char EveButtonFNShift_Caption[6] = "shift";

TCBox_Round *far const code SettingScreen_CBoxes_Round[2]
= {
 &BoxRound2,
```

```
 &BoxRound3
};

TEveKeys *far const code SettingScreen_EveKeys[4] = {
 &EveKeys1,
 &EveKeys2,
 &EveKeys3,
 &EveKeys4
};

TEveButton *far const code SettingScreen_EveButtons[6] = {
 &EveButtonFNfilename,
 &EveButtonFNBackspace,
 &EveButtonFNClear,
 &EveButtonFNConfirm,
 &EveButtonFNGoBack,
 &EveButtonFNShift
};

TCEveText *far const code SettingScreen_CEveTexts[1] = {
 &EveText2
};


TScreen SummaryScreen;

TCBox_Round BoxRound4 = {
 &SummaryScreen,
 0,
 1,
 255,
 255,
 5,
 8,
 469,
 258,
 1,
 0x0000,
 0xD6BA,
 0xE71C,
 3,
 0,
 0,
 0,
 0,
 0
};
TCEveText EveText3 = {
 &SummaryScreen,
 1,
 1,
 255,
 255,
 6,
 10,
 155,
 21,
 EveText3_Caption,
 taNone,
 28,
 0x0148,
 28,
 -1UL,
 1,
 0,
 0,
 0,
 0
};
far const code char EveText3_Caption[21] = "Summary of this
run:";
TCEveText EveText4 = {
 &SummaryScreen,
 2,
 1,
 255,
 255,
 16,
 39,
 100,
 17,
 EveText4_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 1,
 0,
 0,
 0,
 0
};
far const code char EveText4_Caption[17] = "Time Elapsed(s):";
TCEveText EveText5 = {
 &SummaryScreen,
 3,
 1,
 255,
 255,
 16,
 80,
 132,
 17,
 EveText5_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 1,
 0,
 0,
 0,
 0
};
far const code char EveText5_Caption[22] = "Distance
Traveled(m):";
TCEveText EveText6 = {
 &SummaryScreen,
 4,
 1,
 255,
 255,
 16,
 125,
 95,
 17,
 EveText6_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 1,
 0,
 0,
 0,
 0
```

```
};
far const code char EveText6_Caption[16] = "Data File Name:";
TCEveText EveText7 = {
 &SummaryScreen,
 5,
 1,
 255,
 255,
 16,
 206,
 126,
 17,
 EveText7_Caption,
 taNone,
 27,
 0x0148,
 27,
 -1UL,
 1,
 0,
 0,
 0,
 0
};
far const code char EveText7_Caption[20] = "Average
Speed(m/s):";
TCBox_Round BoxRound5 = {
 &SummaryScreen,
 6,
 1,
 255,
 255,
 333,
 75,
 131,
 182,
 1,
 0x0000,
 0xFFFF,
 0xE71C,
 3,
 0,
 0,
 0,
 0,
 0
};
TEveButton EveButtonSUNewRun;
TEvent EveButtonSUNewRun_OnClick;
char EveButtonSUNewRun_Caption[12] = "new profile";
TEveButton EveButtonSUHelp;
TEvent EveButtonSUHelp_OnClick;
char EveButtonSUHelp_Caption[5] = "Help";
TEveButton EveButtonSUAbout;
TEvent EveButtonSUAbout_OnClick;
char EveButtonSUAbout_Caption[6] = "About";
TEveText EveTextSUtime;
char EveTextSUtime_Caption[12] = "0";
TEveText EveTextSUdistance;
char EveTextSUdistance_Caption[12] = "0";
TEveText EveTextSUSpeed;
char EveTextSUSpeed_Caption[15] = "0";
TEveText EveTextSUFilename;
char EveTextSUFilename_Caption[10] = "None";
TEveText EveTextSUAverageSpeed;
char EveTextSUAverageSpeed_Caption[16] = "0";
TEveText EveText8;
char EveText8_Caption[31] = "Last Instantaneous Speed(m/s):";
TBox_Round BoxRound6;
TEveText EveText9;
char EveText9_Caption[9] = "Credits:";
TEveText EveText11;
char EveText11_Caption[55] = "Electronics: Tianyang Chen, Ian
McIntyre, James Weaver";
TEveText EveText13;
char EveText13_Caption[45] = "Mechanical Design: Eric Sinagra,
John Duvall";
TEveText EveText14;
char EveText14_Caption[38] = "Human Engineering Research
Laboratory";
TEveText EveText15;
char EveText15_Caption[25] = "University of Pittsburgh";
TEveText EveText16;
char EveText16_Caption[18] = "VA Medical Center";
TEveText EveText17;
char EveText17_Caption[36] = "US and International Patent
Pending";
TEveButton EveButtonSUBack;
TEvent EveButtonSUBack_OnClick;
char EveButtonSUBack_Caption[5] = "Back";

TBox_Round *far const code SummaryScreen_Boxes_Round[1] =
{
 &BoxRound6
};

TCBox_Round *far const code
SummaryScreen_CBoxes_Round[2] = {
 &BoxRound4,
 &BoxRound5
};

TEveButton *far const code SummaryScreen_EveButtons[4] = {
 &EveButtonSUNewRun,
 &EveButtonSUHelp,
 &EveButtonSUAbout,
 &EveButtonSUBack
};

TEveText *far const code SummaryScreen_EveTexts[13] = {
 &EveTextSUtime,
 &EveTextSUdistance,
 &EveTextSUSpeed,
 &EveTextSUFilename,
 &EveTextSUAverageSpeed,
 &EveText8,
 &EveText9,
 &EveText11,
 &EveText13,
 &EveText14,
 &EveText15,
 &EveText16,
 &EveText17
};

TCEveText *far const code SummaryScreen_CEveTexts[5] = {
 &EveText3,
 &EveText4,
 &EveText5,
 &EveText6,
 &EveText7
};


static char IsInsideObject(TObjInfo *AObjInfo, unsigned int X,
unsigned int Y) {
 TRect *ptrPressRect = 0;
 TRect *ptrPressCircle = 0;
 TCRect *ptrPressRectC = 0;
 TCRect *ptrPressCircleC = 0;
```

```
 if ((AObjInfo->HitX == X) && (AObjInfo->HitY == Y))
 return 1;

 switch (AObjInfo->Type) {

 case VTFT_OT_BOX: {
 ptrPressRect = (TRect *)&(((TBox *)AObjInfo->Obj)->Left);
 break;
 }

 case VTFT_OT_CBOX: {
 ptrPressRectC = (TCRect *)&(((TCBox *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_BOXROUND: {
 ptrPressRect = (TRect *)&(((TBox_Round *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_CBOXROUND: {
 ptrPressRectC = (TCRect *)&(((TCBox_Round *)AObjInfo-
>Obj)->Left);
 break;
 }

 case VTFT_OT_EVEGAUGE: {
 ptrPressCircle = (TRect *)&(((TEveGauge *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_EVEKEYS: {
 ptrPressRect = (TRect *)&(((TEveKeys *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_EVEPROGRESSBAR: {
 ptrPressRect = (TRect *)&(((TEveProgressBar *)AObjInfo-
>Obj)->Left);
 break;
 }

 case VTFT_OT_EVETOGGLE: {
 ptrPressRect = (TRect *)&(((TEveToggle *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_EVEBUTTON: {
 ptrPressRect = (TRect *)&(((TEveButton *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_EVETEXT: {
 ptrPressRect = (TRect *)&(((TEveText *)AObjInfo->Obj)->Left);
 break;
 }

 case VTFT_OT_CEVETEXT: {
 ptrPressRectC = (TCRect *)&(((TCEveText *)AObjInfo->Obj)-
>Left);
 break;
 }

 case VTFT_OT_CEVENUMBER: {
 ptrPressRectC = (TCRect *)&(((TCEveNumber *)AObjInfo-
>Obj)->Left);
 break;
 }
 }

 if (ptrPressRect) {
 if ((ptrPressRect->Left <= X) && (ptrPressRect-
>Left+ptrPressRect->Width-1 >= X) &&
 (ptrPressRect->Top <= Y) && (ptrPressRect->Top+ptrPressRect-
>Height-1 >= Y))
 return 1;
 }
 else if (ptrPressRectC) {
 if ((ptrPressRectC->Left <= X) && (ptrPressRectC-
>Left+ptrPressRectC->Width-1 >= X) &&
 (ptrPressRectC->Top <= Y) && (ptrPressRectC->Top
+ptrPressRectC->Height-1 >= Y))
 return 1;
 }
 else if (ptrPressCircle) {
 if ((ptrPressCircle->Left <= X) && (ptrPressCircle-
>Left+ptrPressCircle->Width*2-1 >= X) &&
 (ptrPressCircle->Top <= Y) && (ptrPressCircle-
>Top+ptrPressCircle->Width*2-1 >= Y))
 return 1;
 }
 else if (ptrPressCircleC) {
 if ((ptrPressCircleC->Left <= X) && (ptrPressCircleC-
>Left+ptrPressCircleC->Width*2-1 >= X) &&
 (ptrPressCircleC->Top <= Y) && (ptrPressCircleC-
>Top+ptrPressCircleC->Width*2-1 >= Y))
 return 1;
 }

 return 0;
}

void DrawBox(TBox *ABox) {
 if (ABox->Visible) {
 if ((VTFT_OT_BOX == TouchS.ActObjInfo.Type) && (ABox
== (TBox *)TouchS.ActObjInfo.Obj)) {
 if (ABox->Gradient != _FT800_BRUSH_GR_NONE) {

FT800_Canvas_BrushGradient(_FT800_BRUSH_STYLE_SOLID
, ABox->Gradient, ABox->Press_Color, ABox->Press_ColorTo,
ABox->Opacity);
 }
 else {

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ABox->Press_Color, ABox->Opacity);
 }
 }
 else {
 if (ABox->Gradient != _FT800_BRUSH_GR_NONE) {

FT800_Canvas_BrushGradient(_FT800_BRUSH_STYLE_SOLID
, ABox->Gradient, ABox->Color, ABox->ColorTo, ABox-
>Opacity);
 }
 else {

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ABox->Color, ABox->Opacity);
 }
 }
```

```
 FT800_Canvas_Pen(ABox->Pen_Width, ABox->Pen_Color,
ABox->Opacity);

 if (ABox->Tag)
 FT800_Canvas_Tag(ABox->Tag);

 FT800_Screen_Box(ABox->Left, ABox->Top, ABox-
>Left+ABox->Width-1, ABox->Top+ABox->Height-1);
 }
}

void DrawCBox(TCBox *ACBox) {
 if (ACBox->Visible) {
 if ((VTFT_OT_CBOX == TouchS.ActObjInfo.Type) &&
(ACBox == (TCBox *)TouchS.ActObjInfo.Obj)) {
 if (ACBox->Gradient != _FT800_BRUSH_GR_NONE) {

FT800_Canvas_BrushGradient(_FT800_BRUSH_STYLE_SOLID
, ACBox->Gradient, ACBox->Press_Color, ACBox-
>Press_ColorTo, ACBox->Opacity);
 }
 else {

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ACBox->Press_Color, ACBox->Opacity);
 }
 }
 else {
 if (ACBox->Gradient != _FT800_BRUSH_GR_NONE) {

FT800_Canvas_BrushGradient(_FT800_BRUSH_STYLE_SOLID
, ACBox->Gradient, ACBox->Color, ACBox->ColorTo, ACBox-
>Opacity);
 }
 else {

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ACBox->Color, ACBox->Opacity);
 }
 }

 FT800_Canvas_Pen(ACBox->Pen_Width, ACBox->Pen_Color,
ACBox->Opacity);

 if (ACBox->Tag)
 FT800_Canvas_Tag(ACBox->Tag);

 FT800_Screen_Box(ACBox->Left, ACBox->Top, ACBox-
>Left+ACBox->Width-1, ACBox->Top+ACBox->Height-1);
 }
}

void DrawBoxRound(TBox_Round *ABoxRound) {
 if (ABoxRound->Visible) {
 if ((VTFT_OT_BOXROUND == TouchS.ActObjInfo.Type) &&
(ABoxRound == (TBox_Round *)TouchS.ActObjInfo.Obj))

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ABoxRound->Press_Color, ABoxRound->Opacity);
 else

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ABoxRound->Color, ABoxRound->Opacity);

 FT800_Canvas_Pen(ABoxRound->Pen_Width, ABoxRound-
>Pen_Color, ABoxRound->Opacity);

 if (ABoxRound->Tag)
 FT800_Canvas_Tag(ABoxRound->Tag);

 FT800_Screen_BoxRound(ABoxRound->Left, ABoxRound-
>Top, ABoxRound->Left+ABoxRound->Width-1, ABoxRound-
>Top+ABoxRound->Height-1, ABoxRound->Corner_Radius);
 }
}

void DrawCBoxRound(TCBox_Round *ACBoxRound) {
 if (ACBoxRound->Visible) {
 if ((VTFT_OT_CBOXROUND == TouchS.ActObjInfo.Type) &&
(ACBoxRound == (TCBox_Round *)TouchS.ActObjInfo.Obj))

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ACBoxRound->Press_Color, ACBoxRound->Opacity);
 else

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, ACBoxRound->Color, ACBoxRound->Opacity);

 FT800_Canvas_Pen(ACBoxRound->Pen_Width, ACBoxRound-
>Pen_Color, ACBoxRound->Opacity);

 if (ACBoxRound->Tag)
 FT800_Canvas_Tag(ACBoxRound->Tag);

 FT800_Screen_BoxRound(ACBoxRound->Left, ACBoxRound-
>Top, ACBoxRound->Left+ACBoxRound->Width-1,
ACBoxRound->Top+ACBoxRound->Height-1, ACBoxRound-
>Corner_Radius);
 }
}

void DrawLine(TLine *ALine) {
 if (ALine->Visible) {
 FT800_Canvas_Pen(ALine->Pen_Width, ALine->Pen_Color,
ALine->Opacity);

 if (ALine->Tag)
 FT800_Canvas_Tag(ALine->Tag);

 FT800_Screen_Line(ALine->First_Point_X, ALine-
>First_Point_Y, ALine->Second_Point_X, ALine-
>Second_Point_Y);
 }
}

void DrawEveGauge(TEveGauge *AEveGauge) {
 unsigned long drawOptions;

 if (AEveGauge->Visible) {
 if ((VTFT_OT_EVEGAUGE == TouchS.ActObjInfo.Type) &&
(AEveGauge == (TEveGauge *)TouchS.ActObjInfo.Obj))

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, AEveGauge->Press_Color, AEveGauge->Opacity);
 else

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, AEveGauge->Color, AEveGauge->Opacity);

 FT800_Canvas_Pen(AEveGauge->Pen_Width, AEveGauge-
>Pen_Color, AEveGauge->Opacity);

 FT800_Canvas_CPGraphics_Major(AEveGauge->Major);
 FT800_Canvas_CPGraphics_Minor(AEveGauge->Minor);

 FT800_Canvas_CPGraphics_Range(AEveGauge->Range);

 FT800_Canvas_CPGraphics_Value(AEveGauge->Value);

 if (AEveGauge->Tag)
```

```
 FT800_Canvas_Tag(AEveGauge->Tag);

 drawOptions = 0;
 if (AEveGauge->Flat)
 drawOptions |= _FT800_CP_DRAW_OPT_FLAT;
 if (AEveGauge->NoBackground)
 drawOptions |= _FT800_CP_DRAW_OPT_NOBACK;
 if (!AEveGauge->TicksVisible)
 drawOptions |= _FT800_CP_DRAW_OPT_NOTICKS;
 if (AEveGauge->NoPointer)
 drawOptions |= _FT800_CP_DRAW_OPT_NOPTR;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_GaugeCp(AEveGauge->Left+AEveGauge-
>Radius,
 AEveGauge->Top+AEveGauge->Radius,
 AEveGauge->Radius);
 }
}

void DrawEveToggle(TEveToggle *AEveToggle) {
 unsigned long drawOptions;

 if (AEveToggle->Visible) {
 if ((VTFT_OT_EVETOGGLE == TouchS.ActObjInfo.Type) &&
(AEveToggle == (TEveToggle *)TouchS.ActObjInfo.Obj))

FT800_Canvas_BrushDualColor(_FT800_BRUSH_STYLE_SOLI
D, AEveToggle->Press_Color, AEveToggle->Color, AEveToggle-
>Opacity);
 else

FT800_Canvas_BrushDualColor(_FT800_BRUSH_STYLE_SOLI
D, AEveToggle->Background_Color, AEveToggle->Color,
AEveToggle->Opacity);

 if (AEveToggle->FontHandle >= 16)
 FT800_Canvas_FontSystem(AEveToggle->FontHandle,
AEveToggle->Font_Color, AEveToggle->Opacity);
 else
 FT800_Canvas_Font(AEveToggle->FontHandle, AEveToggle-
>FontName, AEveToggle->Source, AEveToggle->Font_Color,
AEveToggle->Opacity);

 if (AEveToggle->Tag)
 FT800_Canvas_Tag(AEveToggle->Tag);

 drawOptions = 0;
 if (AEveToggle->Flat)
 drawOptions |= _FT800_CP_DRAW_OPT_FLAT;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_ToggleCP(AEveToggle->Left, AEveToggle-
>Top, AEveToggle->Width, AEveToggle->State, AEveToggle-
>StateOFF_Caption, AEveToggle->StateON_Caption);
 }
}

void DrawEveProgressBar(TEveProgressBar *AEveProgressBar)
{
 unsigned long drawOptions;

 if (AEveProgressBar->Visible) {

FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, AEveProgressBar->Background_Color, AEveProgressBar-
>Opacity);

 FT800_Canvas_Pen(1, AEveProgressBar->Color,
AEveProgressBar->Opacity);

 FT800_Canvas_CPGraphics_Range(AEveProgressBar->Range);

 FT800_Canvas_CPGraphics_Value(AEveProgressBar->Value);

 if (AEveProgressBar->Tag)
 FT800_Canvas_Tag(AEveProgressBar->Tag);

 drawOptions = 0;
 if (AEveProgressBar->Flat)
 drawOptions |= _FT800_CP_DRAW_OPT_FLAT;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_ProgressBarCP(AEveProgressBar->Left,
AEveProgressBar->Top, AEveProgressBar->Width,
AEveProgressBar->Height);
 }
}

void DrawEveKeys(TEveKeys *AEveKeys) {
 unsigned long drawOptions;

 if (AEveKeys->Visible) {
 FT800_Canvas_Brush(_FT800_BRUSH_STYLE_SOLID,
_FT800_BRUSH_GR_BOTTOM_TO_TOP, AEveKeys->Color,
AEveKeys->ColorTo, AEveKeys->Press_Color, AEveKeys-
>Opacity);

 if (AEveKeys->FontHandle >= 16)
 FT800_Canvas_FontSystem(AEveKeys->FontHandle,
AEveKeys->Font_Color, AEveKeys->Opacity);
 else
 FT800_Canvas_Font(AEveKeys->FontHandle, AEveKeys-
>FontName, AEveKeys->Source, AEveKeys->Font_Color,
AEveKeys->Opacity);

 if (AEveKeys->Tag)
 FT800_Canvas_Tag(AEveKeys->Tag);

 drawOptions = 0;
 if (AEveKeys->Flat)
 drawOptions |= _FT800_CP_DRAW_OPT_FLAT;
 if ((VTFT_OT_EVEKEYS == TouchS.ActObjInfo.Type) &&
(AEveKeys == (TEveKeys *)TouchS.ActObjInfo.Obj))
 drawOptions |= TouchS.Tag;
 if (AEveKeys->AutoSize)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTER;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_KeysCP(AEveKeys->Left, AEveKeys->Top,
AEveKeys->Width, AEveKeys->Height, AEveKeys->Caption);
 }
}

void DrawEveButton(TEveButton *AEveButton) {
 unsigned long drawOptions;

 if (AEveButton->Visible) {
 if ((VTFT_OT_EVEBUTTON == TouchS.ActObjInfo.Type) &&
(AEveButton == (TEveButton *)TouchS.ActObjInfo.Obj)) {

FT800_Canvas_BrushGradient(_FT800_BRUSH_STYLE_SOLID
, _FT800_BRUSH_GR_BOTTOM_TO_TOP, AEveButton-
>Press_Color, AEveButton->Press_ColorTo, AEveButton-
>Opacity);
 }
 else {

FT800_Canvas_BrushGradient(_FT800_BRUSH_STYLE_SOLID
```

```
, _FT800_BRUSH_GR_BOTTOM_TO_TOP, AEveButton-
>Color, AEveButton->ColorTo, AEveButton->Opacity);
 }

 if (AEveButton->FontHandle >= 16)
 FT800_Canvas_FontSystem(AEveButton->FontHandle,
AEveButton->Font_Color, AEveButton->Opacity);
 else
 FT800_Canvas_Font(AEveButton->FontHandle, AEveButton-
>FontName, AEveButton->Source, AEveButton->Font_Color,
AEveButton->Opacity);

 if (AEveButton->Tag)
 FT800_Canvas_Tag(AEveButton->Tag);

 drawOptions = 0;
 if (AEveButton->Flat)
 drawOptions |= _FT800_CP_DRAW_OPT_FLAT;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_ButtonCP(AEveButton->Left, AEveButton->Top,
AEveButton->Width, AEveButton->Height, AEveButton-
>Caption);
 }
}

void DrawEveText(TEveText *AEveText) {
 unsigned long drawOptions;

 if (AEveText->Visible) {
 if (AEveText->FontHandle >= 16)
 FT800_Canvas_FontSystem(AEveText->FontHandle, AEveText-
>Font_Color, AEveText->Opacity);
 else
 FT800_Canvas_Font(AEveText->FontHandle, AEveText-
>FontName, AEveText->Source, AEveText->Font_Color,
AEveText->Opacity);

 if (AEveText->Tag)
 FT800_Canvas_Tag(AEveText->Tag);

 drawOptions = 0;
 if (AEveText->TextAlign == taCenter)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTER;
 else if (AEveText->TextAlign == taCenterX)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTERX;
 else if (AEveText->TextAlign == taCenterY)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTERY;
 else if (AEveText->TextAlign == taRightX)
 drawOptions |= _FT800_CP_DRAW_OPT_RIGHTX;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_TextCP(AEveText->Left, AEveText->Top,
AEveText->Caption);
 }
}

void DrawCEveText(TCEveText *ACEveText) {
 unsigned long drawOptions;

 if (ACEveText->Visible) {
 if (ACEveText->FontHandle >= 16)
 FT800_Canvas_FontSystem(ACEveText->FontHandle,
ACEveText->Font_Color, ACEveText->Opacity);
 else
 FT800_Canvas_Font(ACEveText->FontHandle, ACEveText-
>FontName, ACEveText->Source, ACEveText->Font_Color,
ACEveText->Opacity);

 if (ACEveText->Tag)
 FT800_Canvas_Tag(ACEveText->Tag);

 drawOptions = 0;
 if (ACEveText->TextAlign == taCenter)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTER;
 else if (ACEveText->TextAlign == taCenterX)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTERX;
 else if (ACEveText->TextAlign == taCenterY)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTERY;
 else if (ACEveText->TextAlign == taRightX)
 drawOptions |= _FT800_CP_DRAW_OPT_RIGHTX;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_TextCP_Const(ACEveText->Left, ACEveText-
>Top, ACEveText->Caption);
 }
}

void DrawCEveNumber(TCEveNumber *ACEveNumber) {
 unsigned long drawOptions;

 if (ACEveNumber->Visible) {
 if (ACEveNumber->FontHandle >= 16)
 FT800_Canvas_FontSystem(ACEveNumber->FontHandle,
ACEveNumber->Font_Color, ACEveNumber->Opacity);
 else
 FT800_Canvas_Font(ACEveNumber->FontHandle,
ACEveNumber->FontName, ACEveNumber->Source,
ACEveNumber->Font_Color, ACEveNumber->Opacity);

 if (ACEveNumber->Tag)
 FT800_Canvas_Tag(ACEveNumber->Tag);

 drawOptions = 0;
 if (ACEveNumber->TextAlign == taCenter)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTER;
 else if (ACEveNumber->TextAlign == taCenterX)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTERX;
 else if (ACEveNumber->TextAlign == taCenterY)
 drawOptions |= _FT800_CP_DRAW_OPT_CENTERY;
 else if (ACEveNumber->TextAlign == taRightX)
 drawOptions |= _FT800_CP_DRAW_OPT_RIGHTX;
 if (ACEveNumber->Signed)
 drawOptions |= _FT800_CP_DRAW_OPT_SIGNED;
 drawOptions |= ACEveNumber->Text_Length;
 FT800_Canvas_CPGraphics_DrawingOptions(drawOptions);

 FT800_Screen_NumberCP(ACEveNumber->Left,
ACEveNumber->Top, ACEveNumber->Value);
 }
}

void DrawObject(TPointer aObj, char aObjType) {
 TDrawHandler drawHandler;

 drawHandler = DrawHandlerTable[aObjType];
 if (drawHandler)
 drawHandler(aObj);
}

void DrawScreenO(TScreen *aScreen, char aOptions) {
 unsigned short cOrder, saveOrder;
 signed int actObjOrder;
 unsigned short pwmDuty;

 char cntBox;
 char cntCBox;
 char cntBoxRound;
 char cntCBoxRound;
 char cntLine;
```

```
char cntEveGauge;
char cntEveKeys;
char cntEveProgressBar;
char cntEveToggle;
char cntEveButton;
char cntEveText;
char cntCEveText;
char cntCEveNumber;

TBox *far const code *pBox;
TCBox *far const code *pCBox;
TBox_Round *far const code *pBoxRound;
TCBox_Round *far const code *pCBoxRound;
TLine *far const code *pLine;
TEveGauge *far const code *pEveGauge;
TEveKeys *far const code *pEveKeys;
TEveProgressBar *far const code *pEveProgressBar;
TEveToggle *far const code *pEveToggle;
TEveButton *far const code *pEveButton;
TEveText *far const code *pEveText;
TCEveText *far const code *pCEveText;
TCEveNumber *far const code *pCEveNumber;


if (aOptions & VTFT_DISPLAY_EFF_LIGHTS_FADE) {
FT800_PWM_Get(0, &pwmDuty);
FT800_PWM_FadeOut(pwmDuty, 0, pwmDuty/32? pwmDuty/32
: 1, 1);
 }
else if (aOptions & VTFT_DISPLAY_EFF_LIGHTS_OFF) {
FT800_PWM_Get(0, &pwmDuty);
FT800_PWM_Duty(0);
 }


if (CurrentScreen != aScreen)
memset(&TouchS.ActObjInfo, 0, sizeof(TObjInfo));

CurrentScreen = aScreen;


cntBox = CurrentScreen->BoxesCount;
cntCBox = CurrentScreen->CBoxesCount;
cntBoxRound = CurrentScreen->Boxes_RoundCount;
cntCBoxRound = CurrentScreen->CBoxes_RoundCount;
cntLine = CurrentScreen->LinesCount;
cntEveGauge = CurrentScreen->EveGaugesCount;
cntEveKeys = CurrentScreen->EveKeysCount;
cntEveProgressBar = CurrentScreen->EveProgressBarsCount;
cntEveToggle = CurrentScreen->EveTogglesCount;
cntEveButton = CurrentScreen->EveButtonsCount;
cntEveText = CurrentScreen->EveTextsCount;
cntCEveText = CurrentScreen->CEveTextsCount;
cntCEveNumber = CurrentScreen->CEveNumbersCount;

pBox = CurrentScreen->Boxes;
pCBox = CurrentScreen->CBoxes;
pBoxRound = CurrentScreen->Boxes_Round;
pCBoxRound = CurrentScreen->CBoxes_Round;
pLine = CurrentScreen->Lines;
pEveGauge = CurrentScreen->EveGauges;
pEveKeys = CurrentScreen->EveKeys;
pEveProgressBar = CurrentScreen->EveProgressBars;
pEveToggle = CurrentScreen->EveToggles;
pEveButton = CurrentScreen->EveButtons;
pEveText = CurrentScreen->EveTexts;
pCEveText = CurrentScreen->CEveTexts;
pCEveNumber = CurrentScreen->CEveNumbers;

FT800_Screen_BeginUpdateCP();
FT800_Canvas_BrushSingleColor(_FT800_BRUSH_STYLE_SO
LID, CurrentScreen->Color, 255);
 FT800_Canvas_Tag(0);
 FT800_Screen_Clear(_FT800_CLEAR_ALL);

 actObjOrder = -1;
 if (TouchS.ActObjInfo.Obj)
 if (TouchS.ActObjInfo.Flags &
VTFT_INT_BRING_TO_FRONT)
 actObjOrder = TouchS.ActObjInfo.Order;

 cOrder = 0;
 while (cOrder < CurrentScreen->ObjectsCount) {
 saveOrder = cOrder;
 if (pBox) {
 while ((*pBox)->Order == cOrder) {
 if (actObjOrder != cOrder)
 DrawBox(*pBox);
 cOrder++;
 pBox++;
 cntBox--;
 if (!cntBox) {
 pBox = 0;
 break;
 }
 }
 if (saveOrder != cOrder)
 continue;
 }

 if (pCBox) {
 while ((*pCBox)->Order == cOrder) {
 if (actObjOrder != cOrder)
 DrawCBox(*pCBox);
 cOrder++;
 pCBox++;
 cntCBox--;
 if (!cntCBox) {
 pCBox = 0;
 break;
 }
 }
 if (saveOrder != cOrder)
 continue;
 }

 if (pBoxRound) {
 while ((*pBoxRound)->Order == cOrder) {
 if (actObjOrder != cOrder)
 DrawBoxRound(*pBoxRound);
 cOrder++;
 pBoxRound++;
 cntBoxRound--;
 if (!cntBoxRound) {
 pBoxRound = 0;
 break;
 }
 }
 if (saveOrder != cOrder)
 continue;
 }

 if (pCBoxRound) {
 while ((*pCBoxRound)->Order == cOrder) {
 if (actObjOrder != cOrder)
 DrawCBoxRound(*pCBoxRound);
 cOrder++;
 pCBoxRound++;
 cntCBoxRound--;
```

```
if (!cntCBoxRound) {
pCBoxRound = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pLine) {
while ((*pLine)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawLine(*pLine);
cOrder++;
pLine++;
cntLine--;
if (!cntLine) {
pLine = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pEveGauge) {
while ((*pEveGauge)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawEveGauge(*pEveGauge);
cOrder++;
pEveGauge++;
cntEveGauge--;
if (!cntEveGauge) {
pEveGauge = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pEveKeys) {
while ((*pEveKeys)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawEveKeys(*pEveKeys);
cOrder++;
pEveKeys++;
cntEveKeys--;
if (!cntEveKeys) {
pEveKeys = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pEveProgressBar) {
while ((*pEveProgressBar)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawEveProgressBar(*pEveProgressBar);
cOrder++;
pEveProgressBar++;
cntEveProgressBar--;
if (!cntEveProgressBar) {
pEveProgressBar = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pEveToggle) {
while ((*pEveToggle)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawEveToggle(*pEveToggle);
cOrder++;
pEveToggle++;
cntEveToggle--;
if (!cntEveToggle) {
pEveToggle = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pEveButton) {
while ((*pEveButton)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawEveButton(*pEveButton);
cOrder++;
pEveButton++;
cntEveButton--;
if (!cntEveButton) {
pEveButton = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pEveText) {
while ((*pEveText)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawEveText(*pEveText);
cOrder++;
pEveText++;
cntEveText--;
if (!cntEveText) {
pEveText = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pCEveText) {
while ((*pCEveText)->Order == cOrder) {
if (actObjOrder != cOrder)
DrawCEveText(*pCEveText);
cOrder++;
pCEveText++;
cntCEveText--;
if (!cntCEveText) {
pCEveText = 0;
break;
}
}
if (saveOrder != cOrder)
continue;
}

if (pCEveNumber) {
while ((*pCEveNumber)->Order == cOrder) {
if (actObjOrder != cOrder)
```

```
 DrawCEveNumber(*pCEveNumber);
 cOrder++;
 pCEveNumber++;
 cntCEveNumber--;
 if (!cntCEveNumber) {
 pCEveNumber = 0;
 break;
 }
 }
 if (saveOrder != cOrder)
 continue;
 }

 cOrder++;
 }


 if (TouchS.ActObjInfo.Obj)
 DrawObject(TouchS.ActObjInfo.Obj, TouchS.ActObjInfo.Type);

 FT800_Screen_EndUpdate();

 FT800_Screen_Show();


 if (aOptions & VTFT_DISPLAY_EFF_LIGHTS_FADE) {
 FT800_PWM_FadeIn(0, pwmDuty, 1, 3);
 }
 else if (aOptions & VTFT_DISPLAY_EFF_LIGHTS_OFF) {
 FT800_PWM_Duty(pwmDuty);
 }

}

void DrawScreen(TScreen *aScreen) {
 if (aScreen != CurrentScreen)
 DrawScreenO(aScreen, VTFT_LOAD_RES_ALL |
VTFT_DISPLAY_EFF_LIGHTS_FADE);
 else
 DrawScreenO(aScreen, VTFT_LOAD_RES_NONE);
}

char GetActiveObjectByXY(int X, int Y, TObjInfo *AObjInfo) {
 char i;
 int hiOrder;
 TBox *pBox;
 TCBox *pCBox;
 TBox_Round *pBoxRound;
 TCBox_Round *pCBoxRound;
 TEveGauge *pEveGauge;
 TEveKeys *pEveKeys;
 TEveProgressBar *pEveProgressBar;
 TEveToggle *pEveToggle;
 TEveButton *pEveButton;
 TEveText *pEveText;
 TCEveText *pCEveText;
 TCEveNumber *pCEveNumber;
 far const code void *far const code *ptrCO;
 void *far const code *ptrO;


 memset(AObjInfo, 0, sizeof(TObjInfo));



 hiOrder = -1;


 i = CurrentScreen->BoxesCount;
 ptrO = CurrentScreen->Boxes+CurrentScreen->BoxesCount-1;
 while (i--) {
 pBox = (TBox *)(*ptrO);
 if (pBox->Order < hiOrder)
 break;
 if (pBox->Active) {
 if ((pBox->Left <= X) && (pBox->Left+pBox->Width-1 >= X)
&&
 (pBox->Top <= Y) && (pBox->Top+pBox->Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pBox;
 AObjInfo->Type = VTFT_OT_BOX;
 AObjInfo->Order = pBox->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pBox->Press_Color != pBox->Color) ||
 ((pBox->Gradient != _FT800_BRUSH_GR_NONE) &&
 (pBox->Press_ColorTo != pBox->ColorTo)))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pBox->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->CBoxesCount;
 ptrCO = CurrentScreen->CBoxes+CurrentScreen->CBoxesCount-
1;
 while (i--) {
 pCBox = (TCBox *)(*ptrCO);
 if (pCBox->Order < hiOrder)
 break;
 if (pCBox->Active) {
 if ((pCBox->Left <= X) && (pCBox->Left+pCBox->Width-1 >=
X) &&
 (pCBox->Top <= Y) && (pCBox->Top+pCBox->Height-1 >=
Y)) {
 AObjInfo->Obj = (TPointer)pCBox;
 AObjInfo->Type = VTFT_OT_CBOX;
 AObjInfo->Order = pCBox->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pCBox->Press_Color != pCBox->Color) ||
 ((pCBox->Gradient != _FT800_BRUSH_GR_NONE) &&
 (pCBox->Press_ColorTo != pCBox->ColorTo)))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pCBox->Order;

 break;
 }
 }
 ptrCO--;
 }


 i = CurrentScreen->Boxes_RoundCount;
 ptrO = CurrentScreen->Boxes_Round+CurrentScreen-
>Boxes_RoundCount-1;
 while (i--) {
 pBoxRound = (TBox_Round *)(*ptrO);
 if (pBoxRound->Order < hiOrder)
 break;
 if (pBoxRound->Active) {
 if ((pBoxRound->Left <= X) && (pBoxRound-
>Left+pBoxRound->Width-1 >= X) &&
 (pBoxRound->Top <= Y) && (pBoxRound->Top+pBoxRound-
>Height-1 >= Y)) {
```

```
 AObjInfo->Obj = (TPointer)pBoxRound;
 AObjInfo->Type = VTFT_OT_BOXROUND;
 AObjInfo->Order = pBoxRound->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if (pBoxRound->Press_Color != pBoxRound->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pBoxRound->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->CBoxes_RoundCount;
 ptrCO = CurrentScreen->CBoxes_Round+CurrentScreen-
>CBoxes_RoundCount-1;
 while (i--) {
 pCBoxRound = (TCBox_Round *)(*ptrCO);
 if (pCBoxRound->Order < hiOrder)
 break;
 if (pCBoxRound->Active) {
 if ((pCBoxRound->Left <= X) && (pCBoxRound-
>Left+pCBoxRound->Width-1 >= X) &&
 (pCBoxRound->Top <= Y) && (pCBoxRound-
>Top+pCBoxRound->Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pCBoxRound;
 AObjInfo->Type = VTFT_OT_CBOXROUND;
 AObjInfo->Order = pCBoxRound->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if (pCBoxRound->Press_Color != pCBoxRound->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pCBoxRound->Order;

 break;
 }
 }
 ptrCO--;
 }


 i = CurrentScreen->EveGaugesCount;
 ptrO = CurrentScreen->EveGauges+CurrentScreen-
>EveGaugesCount-1;
 while (i--) {
 pEveGauge = (TEveGauge *)(*ptrO);
 if (pEveGauge->Order < hiOrder)
 break;
 if (pEveGauge->Active) {

 if ((pEveGauge->Left <= X) && (pEveGauge->Left+pEveGauge-
>Radius*2-1 >= X) &&
 (pEveGauge->Top <= Y) && (pEveGauge->Top+pEveGauge-
>Radius*2-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pEveGauge;
 AObjInfo->Type = VTFT_OT_EVEGAUGE;
 AObjInfo->Order = pEveGauge->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if (pEveGauge->Press_Color != pEveGauge->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pEveGauge->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->EveKeysCount;
 ptrO = CurrentScreen->EveKeys+CurrentScreen-
>EveKeysCount-1;
 while (i--) {
 pEveKeys = (TEveKeys *)(*ptrO);
 if (pEveKeys->Order < hiOrder)
 break;
 if (pEveKeys->Active) {
 if ((pEveKeys->Left <= X) && (pEveKeys->Left+pEveKeys-
>Width-1 >= X) &&
 (pEveKeys->Top <= Y) && (pEveKeys->Top+pEveKeys-
>Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pEveKeys;
 AObjInfo->Type = VTFT_OT_EVEKEYS;
 AObjInfo->Order = pEveKeys->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pEveKeys->Press_Color != pEveKeys->Color) ||
 (pEveKeys->Press_ColorTo != pEveKeys->ColorTo))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pEveKeys->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->EveTogglesCount;
 ptrO = CurrentScreen->EveToggles+CurrentScreen-
>EveTogglesCount-1;
 while (i--) {
 pEveToggle = (TEveToggle *)(*ptrO);
 if (pEveToggle->Order < hiOrder)
 break;
 if (pEveToggle->Active) {
 if ((pEveToggle->Left <= X) && (pEveToggle-
>Left+pEveToggle->Width-1 >= X) &&
 (pEveToggle->Top <= Y) && (pEveToggle->Top+pEveToggle-
>Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pEveToggle;
 AObjInfo->Type = VTFT_OT_EVETOGGLE;
 AObjInfo->Order = pEveToggle->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 AObjInfo->Flags |= VTFT_INT_INTRINSIC_CLICK_EFF;
 if (pEveToggle->Background_Color != pEveToggle->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pEveToggle->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->EveButtonsCount;
 ptrO = CurrentScreen->EveButtons+CurrentScreen-
>EveButtonsCount-1;
 while (i--) {
```

```
 pEveButton = (TEveButton *)(*ptrO);
 if (pEveButton->Order < hiOrder)
 break;
 if (pEveButton->Active) {
 if ((pEveButton->Left <= X) && (pEveButton-
>Left+pEveButton->Width-1 >= X) &&
 (pEveButton->Top <= Y) && (pEveButton->Top+pEveButton-
>Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pEveButton;
 AObjInfo->Type = VTFT_OT_EVEBUTTON;
 AObjInfo->Order = pEveButton->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pEveButton->Press_Color != pEveButton->Color) ||
 (pEveButton->Press_ColorTo != pEveButton->ColorTo))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 hiOrder = pEveButton->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->EveTextsCount;
 ptrO = CurrentScreen->EveTexts+CurrentScreen-
>EveTextsCount-1;
 while (i--) {
 pEveText = (TEveText *)(*ptrO);
 if (pEveText->Order < hiOrder)
 break;
 if (pEveText->Active) {
 if ((pEveText->Left <= X) && (pEveText->Left+pEveText-
>Width-1 >= X) &&
 (pEveText->Top <= Y) && (pEveText->Top+pEveText->Height-
1 >= Y)) {
 AObjInfo->Obj = (TPointer)pEveText;
 AObjInfo->Type = VTFT_OT_EVETEXT;
 AObjInfo->Order = pEveText->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;

 hiOrder = pEveText->Order;

 break;
 }
 }
 ptrO--;
 }


 i = CurrentScreen->CEveTextsCount;
 ptrCO = CurrentScreen->CEveTexts+CurrentScreen-
>CEveTextsCount-1;
 while (i--) {
 pCEveText = (TCEveText *)(*ptrCO);
 if (pCEveText->Order < hiOrder)
 break;
 if (pCEveText->Active) {
 if ((pCEveText->Left <= X) && (pCEveText->Left+pCEveText-
>Width-1 >= X) &&
 (pCEveText->Top <= Y) && (pCEveText->Top+pCEveText-
>Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pCEveText;
 AObjInfo->Type = VTFT_OT_CEVETEXT;
 AObjInfo->Order = pCEveText->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;

 hiOrder = pCEveText->Order;

 break;
 }
 }
 ptrCO--;
 }


 i = CurrentScreen->CEveNumbersCount;
 ptrCO = CurrentScreen->CEveNumbers+CurrentScreen-
>CEveNumbersCount-1;
 while (i--) {
 pCEveNumber = (TCEveNumber *)(*ptrCO);
 if (pCEveNumber->Order < hiOrder)
 break;
 if (pCEveNumber->Active) {
 if ((pCEveNumber->Left <= X) && (pCEveNumber-
>Left+pCEveNumber->Width-1 >= X) &&
 (pCEveNumber->Top <= Y) && (pCEveNumber-
>Top+pCEveNumber->Height-1 >= Y)) {
 AObjInfo->Obj = (TPointer)pCEveNumber;
 AObjInfo->Type = VTFT_OT_CEVENUMBER;
 AObjInfo->Order = pCEveNumber->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;

 hiOrder = pCEveNumber->Order;

 break;
 }
 }
 ptrCO--;
 }

 if (AObjInfo->Obj) {
 AObjInfo->HitX = X;
 AObjInfo->HitY = Y;
 return 1;
 }
 else {
 return 0;
 }
}

char GetActiveObjectByTag(char ATag, TObjInfo *AObjInfo) {
 char i;
 TBox *pBox;
 TCBox *pCBox;
 TBox_Round *pBoxRound;
 TCBox_Round *pCBoxRound;
 TEveGauge *pEveGauge;
 TEveKeys *pEveKeys;
 TEveProgressBar *pEveProgressBar;
 TEveToggle *pEveToggle;
 TEveButton *pEveButton;
 TEveText *pEveText;
 TCEveText *pCEveText;
 TCEveNumber *pCEveNumber;
 far const code void *far const code *ptrCO;
 void *far const code *ptrO;


 memset(AObjInfo, 0, sizeof(TObjInfo));



 i = CurrentScreen->BoxesCount;
 ptrO = CurrentScreen->Boxes+CurrentScreen->BoxesCount-1;
 while (i--) {
 pBox = (TBox *)(*ptrO);
```

```
 if (pBox->Tag == ATag) {
 if (pBox->Active) {
 AObjInfo->Obj = (TPointer)pBox;
 AObjInfo->Type = VTFT_OT_BOX;
 AObjInfo->Order = pBox->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pBox->Press_Color != pBox->Color) ||
 ((pBox->Gradient != _FT800_BRUSH_GR_NONE) &&
 (pBox->Press_ColorTo != pBox->ColorTo)))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->CBoxesCount;
 ptrCO = CurrentScreen->CBoxes+CurrentScreen->CBoxesCount-
1;
 while (i--) {
 pCBox = (TCBox *)(*ptrCO);
 if (pCBox->Tag == ATag) {
 if (pCBox->Active) {
 AObjInfo->Obj = (TPointer)pCBox;
 AObjInfo->Type = VTFT_OT_CBOX;
 AObjInfo->Order = pCBox->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pCBox->Press_Color != pCBox->Color) ||
 ((pCBox->Gradient != _FT800_BRUSH_GR_NONE) &&
 (pCBox->Press_ColorTo != pCBox->ColorTo)))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrCO--;
 }


 i = CurrentScreen->Boxes_RoundCount;
 ptrO = CurrentScreen->Boxes_Round+CurrentScreen-
>Boxes_RoundCount-1;
 while (i--) {
 pBoxRound = (TBox_Round *)(*ptrO);
 if (pBoxRound->Tag == ATag) {
 if (pBoxRound->Active) {
 AObjInfo->Obj = (TPointer)pBoxRound;
 AObjInfo->Type = VTFT_OT_BOXROUND;
 AObjInfo->Order = pBoxRound->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if (pBoxRound->Press_Color != pBoxRound->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->CBoxes_RoundCount;
 ptrCO = CurrentScreen->CBoxes_Round+CurrentScreen-
>CBoxes_RoundCount-1;
 while (i--) {
 pCBoxRound = (TCBox_Round *)(*ptrCO);
 if (pCBoxRound->Tag == ATag) {
 if (pCBoxRound->Active) {
 AObjInfo->Obj = (TPointer)pCBoxRound;
 AObjInfo->Type = VTFT_OT_CBOXROUND;
 AObjInfo->Order = pCBoxRound->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if (pCBoxRound->Press_Color != pCBoxRound->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrCO--;
 }


 i = CurrentScreen->EveGaugesCount;
 ptrO = CurrentScreen->EveGauges+CurrentScreen-
>EveGaugesCount-1;
 while (i--) {
 pEveGauge = (TEveGauge *)(*ptrO);
 if (pEveGauge->Tag == ATag) {
 if (pEveGauge->Active) {
 AObjInfo->Obj = (TPointer)pEveGauge;
 AObjInfo->Type = VTFT_OT_EVEGAUGE;
 AObjInfo->Order = pEveGauge->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if (pEveGauge->Press_Color != pEveGauge->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->EveKeysCount;
 ptrO = CurrentScreen->EveKeys+CurrentScreen-
>EveKeysCount-1;
 while (i--) {
 pEveKeys = (TEveKeys *)(*ptrO);
 if (pEveKeys->Tag == ATag) {
 if (pEveKeys->Active) {
 AObjInfo->Obj = (TPointer)pEveKeys;
 AObjInfo->Type = VTFT_OT_EVEKEYS;
 AObjInfo->Order = pEveKeys->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pEveKeys->Press_Color != pEveKeys->Color) ||
 (pEveKeys->Press_ColorTo != pEveKeys->ColorTo))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->EveTogglesCount;
 ptrO = CurrentScreen->EveToggles+CurrentScreen-
>EveTogglesCount-1;
 while (i--) {
 pEveToggle = (TEveToggle *)(*ptrO);
 if (pEveToggle->Tag == ATag) {
 if (pEveToggle->Active) {
 AObjInfo->Obj = (TPointer)pEveToggle;
```

```
 AObjInfo->Type = VTFT_OT_EVETOGGLE;
 AObjInfo->Order = pEveToggle->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 AObjInfo->Flags |= VTFT_INT_INTRINSIC_CLICK_EFF;
 if (pEveToggle->Background_Color != pEveToggle->Color)
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->EveButtonsCount;
 ptrO = CurrentScreen->EveButtons+CurrentScreen-
>EveButtonsCount-1;
 while (i--) {
 pEveButton = (TEveButton *)(*ptrO);
 if (pEveButton->Tag == ATag) {
 if (pEveButton->Active) {
 AObjInfo->Obj = (TPointer)pEveButton;
 AObjInfo->Type = VTFT_OT_EVEBUTTON;
 AObjInfo->Order = pEveButton->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 if ((pEveButton->Press_Color != pEveButton->Color) ||
 (pEveButton->Press_ColorTo != pEveButton->ColorTo))
 AObjInfo->Flags |= VTFT_INT_REPAINT_ON_DOWN |
VTFT_INT_REPAINT_ON_UP;

 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->EveTextsCount;
 ptrO = CurrentScreen->EveTexts+CurrentScreen-
>EveTextsCount-1;
 while (i--) {
 pEveText = (TEveText *)(*ptrO);
 if (pEveText->Tag == ATag) {
 if (pEveText->Active) {
 AObjInfo->Obj = (TPointer)pEveText;
 AObjInfo->Type = VTFT_OT_EVETEXT;
 AObjInfo->Order = pEveText->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 }
 break;
 }
 ptrO--;
 }


 i = CurrentScreen->CEveTextsCount;
 ptrCO = CurrentScreen->CEveTexts+CurrentScreen-
>CEveTextsCount-1;
 while (i--) {
 pCEveText = (TCEveText *)(*ptrCO);
 if (pCEveText->Tag == ATag) {
 if (pCEveText->Active) {
 AObjInfo->Obj = (TPointer)pCEveText;
 AObjInfo->Type = VTFT_OT_CEVETEXT;
 AObjInfo->Order = pCEveText->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 }
 break;
 }
 ptrCO--;
 }


 i = CurrentScreen->CEveNumbersCount;
 ptrCO = CurrentScreen->CEveNumbers+CurrentScreen-
>CEveNumbersCount-1;
 while (i--) {
 pCEveNumber = (TCEveNumber *)(*ptrCO);
 if (pCEveNumber->Tag == ATag) {
 if (pCEveNumber->Active) {
 AObjInfo->Obj = (TPointer)pCEveNumber;
 AObjInfo->Type = VTFT_OT_CEVENUMBER;
 AObjInfo->Order = pCEveNumber->Order;
 AObjInfo->Flags = VTFT_INT_BRING_TO_FRONT;
 }
 break;
 }
 ptrCO--;
 }

 if (AObjInfo->Obj) {
 AObjInfo->HitTag = ATag;
 return 1;
 }
 else {
 return 0;
 }
}

static void ProcessEvent(TEvent *pEvent) {
 if (pEvent) {
 if (pEvent->Sound.SndAct == VTFT_SNDACT_PLAY)
 FT800_Sound_SetAndPlay(pEvent->Sound.Effect, pEvent-
>Sound.Pitch, pEvent->Sound.Volume);
 else if (pEvent->Sound.SndAct == VTFT_SNDACT_STOP)
 FT800_Sound_Stop();
 if (pEvent->Action)
 pEvent->Action();
 }
}

static void ProcessCEvent(TCEvent *pEventC) {
 if (pEventC) {
 if (pEventC->Sound.SndAct == VTFT_SNDACT_PLAY)
 FT800_Sound_SetAndPlay(pEventC->Sound.Effect, pEventC-
>Sound.Pitch, pEventC->Sound.Volume);
 else if (pEventC->Sound.SndAct == VTFT_SNDACT_STOP)
 FT800_Sound_Stop();
 if (pEventC->Action)
 pEventC->Action();
 }
}

static void OnEvent(TObjInfo *AObjInfo, char AEventType){
 TEvent **ppEvent;
 TEvent *pEvent = 0;
 TCEvent *far const code *ppEventC;
 TCEvent *pEventC = 0;

 switch (AObjInfo->Type) {

 case VTFT_OT_BOX: {
 ppEvent = &(((TBox *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_CBOX: {
 ppEventC = &(((TCBox *)AObjInfo->Obj)->OnUp);
```

```
 pEventC = ppEventC[AEventType];
 break;
 }

 case VTFT_OT_BOXROUND: {
 ppEvent = &(((TBox_Round *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_CBOXROUND: {
 ppEventC = &(((TCBox_Round *)AObjInfo->Obj)->OnUp);
 pEventC = ppEventC[AEventType];
 break;
 }

 case VTFT_OT_EVEGAUGE: {
 ppEvent = &(((TEveGauge *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_EVEKEYS: {
 ppEvent = &(((TEveKeys *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_EVETOGGLE: {
 ppEvent = &(((TEveToggle *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_EVEBUTTON: {
 ppEvent = &(((TEveButton *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_EVETEXT: {
 ppEvent = &(((TEveText *)AObjInfo->Obj)->OnUp);
 pEvent = ppEvent[AEventType];
 break;
 }

 case VTFT_OT_CEVETEXT: {
 ppEventC = &(((TCEveText *)AObjInfo->Obj)->OnUp);
 pEventC = ppEventC[AEventType];
 break;
 }

 case VTFT_OT_CEVENUMBER: {
 ppEventC = &(((TCEveNumber *)AObjInfo->Obj)->OnUp);
 pEventC = ppEventC[AEventType];
 break;
 }
 }

 if (pEvent) {
 ProcessEvent(pEvent);
 }

 if (pEventC) {
 ProcessCEvent(pEventC);
 }
}

static void ProcessIntrinsicClickEffects(TObjInfo *AObjInfo) {
 char i;
 int toggleStepVal;
 const char TOGGLE_STEP_CNT = 15;

 if (!(AObjInfo->Flags & VTFT_INT_INTRINSIC_CLICK_EFF))
 return;

 switch (AObjInfo->Type) {
 case VTFT_OT_EVETOGGLE:
 {
 toggleStepVal = 65535/TOGGLE_STEP_CNT;
 if (((TEveToggle *)AObjInfo->Obj)->State)
 {
 toggleStepVal = -toggleStepVal;
 }
 for (i=0; i<TOGGLE_STEP_CNT; i++)
 {
 ((TEveToggle *)AObjInfo->Obj)->State += toggleStepVal;
 DrawScreen(CurrentScreen);
 }
 AObjInfo->Flags &= ~VTFT_INT_REPAINT_ON_UP;
 break;
 }
 }
}

static void Process_TP_Press() {

 if (CurrentScreen->Active)
 if ((CurrentScreen->SniffObjectEvents) ||
(!TouchS.ActObjInfo.Obj))
 ProcessEvent(CurrentScreen->OnPress);


 if (!TouchS.ActObjInfo.Obj)
 return;

 OnEvent(&TouchS.ActObjInfo, VTFT_EVT_PRESS);
}

static void Process_TP_Up() {
 char isClick;
 TObjInfo actObj;


 if (CurrentScreen->Active)
 if ((CurrentScreen->SniffObjectEvents) ||
(!TouchS.ActObjInfo.Obj))
 ProcessEvent(CurrentScreen->OnUp);

 actObj = TouchS.ActObjInfo;

 memset(&TouchS.ActObjInfo, 0, sizeof(TObjInfo));


 if (!actObj.Obj)
 return;

 isClick = IsInsideObject(&actObj, TouchS.X, TouchS.Y);

 if (isClick) {
 ProcessIntrinsicClickEffects(&actObj);
 }

 if (actObj.Flags & VTFT_INT_REPAINT_ON_UP)
 DrawScreen(CurrentScreen);

 OnEvent(&actObj, VTFT_EVT_UP);
 if (isClick)
 OnEvent(&actObj, VTFT_EVT_CLICK);
}
```

```
static void Process_TP_Down() {

 if (TouchS.Tag) {
 if (TouchS.Tag != 255)
 GetActiveObjectByTag(TouchS.Tag, &TouchS.ActObjInfo);
 if (!TouchS.ActObjInfo.Obj)
 GetActiveObjectByXY(TouchS.X, TouchS.Y,
&TouchS.ActObjInfo);
 }


 if (CurrentScreen->Active)
 if ((CurrentScreen->SniffObjectEvents) ||
(!TouchS.ActObjInfo.Obj))
 ProcessEvent(CurrentScreen->OnDown);


 if (!TouchS.ActObjInfo.Obj)
 return;

 if (TouchS.ActObjInfo.Flags &
VTFT_INT_REPAINT_ON_DOWN)
 DrawScreen(CurrentScreen);

 OnEvent(&TouchS.ActObjInfo, VTFT_EVT_DOWN);
}

static void Process_TP_TagChange() {

 if (CurrentScreen->Active)
 ProcessEvent(CurrentScreen->OnTagChange);
}

void ProcessVTFTStack() {
 char Tag, oldTag;
 unsigned int X, Y;

 oldTag = TouchS.Tag;

 if (FT800_Touch_GetTagXY(&X, &Y) == 1) {
 FT800_Touch_GetTag(&Tag);

 TouchS.Tag = Tag;
 TouchS.X = X;
 TouchS.Y = Y;

 if (!TouchS.Pressed) {
 TouchS.Pressed = 1;
 Process_TP_Down();
 }

 Process_TP_Press();
 }
 else if (TouchS.Pressed) {
 Process_TP_Up();

 TouchS.Tag = 0;
 TouchS.X = X;
 TouchS.Y = Y;

 TouchS.Pressed = 0;
 }

 if (oldTag != TouchS.Tag)
 Process_TP_TagChange();
}

static void InitObjects() {

 SplashScreen.Color = 0xFFFF;
 SplashScreen.Width = 480;
 SplashScreen.Height = 272;
 SplashScreen.ObjectsCount = 4;
 SplashScreen.BoxesCount = 0;
 SplashScreen.Boxes = 0;
 SplashScreen.CBoxesCount = 0;
 SplashScreen.CBoxes = 0;
 SplashScreen.Boxes_RoundCount = 0;
 SplashScreen.Boxes_Round = 0;
 SplashScreen.CBoxes_RoundCount = 0;
 SplashScreen.CBoxes_Round = 0;
 SplashScreen.LinesCount = 0;
 SplashScreen.Lines = 0;
 SplashScreen.EveGaugesCount = 0;
 SplashScreen.EveGauges = 0;
 SplashScreen.EveKeysCount = 0;
 SplashScreen.EveKeys = 0;
 SplashScreen.EveProgressBarsCount = 1;
 SplashScreen.EveProgressBars = SplashScreen_EveProgressBars;
 SplashScreen.EveTogglesCount = 0;
 SplashScreen.EveToggles = 0;
 SplashScreen.EveButtonsCount = 1;
 SplashScreen.EveButtons = SplashScreen_EveButtons;
 SplashScreen.EveTextsCount = 1;
 SplashScreen.EveTexts = SplashScreen_EveTexts;
 SplashScreen.CEveTextsCount = 1;
 SplashScreen.CEveTexts = SplashScreen_CEveTexts;
 SplashScreen.CEveNumbersCount = 0;
 SplashScreen.CEveNumbers = 0;
 SplashScreen.DynResStart = 0;
 SplashScreen.Active = 0;
 SplashScreen.SniffObjectEvents = 0;
 SplashScreen.OnUp = 0;
 SplashScreen.OnDown = 0;
 SplashScreen.OnTagChange = 0;
 SplashScreen.OnPress = 0;


 SamplingScreen.Color = 0xFFFF;
 SamplingScreen.Width = 480;
 SamplingScreen.Height = 272;
 SamplingScreen.ObjectsCount = 27;
 SamplingScreen.BoxesCount = 0;
 SamplingScreen.Boxes = 0;
 SamplingScreen.CBoxesCount = 0;
 SamplingScreen.CBoxes = 0;
 SamplingScreen.Boxes_RoundCount = 2;
 SamplingScreen.Boxes_Round = SamplingScreen_Boxes_Round;
 SamplingScreen.CBoxes_RoundCount = 2;
 SamplingScreen.CBoxes_Round =
SamplingScreen_CBoxes_Round;
 SamplingScreen.LinesCount = 0;
 SamplingScreen.Lines = 0;
 SamplingScreen.EveGaugesCount = 0;
 SamplingScreen.EveGauges = 0;
 SamplingScreen.EveKeysCount = 0;
 SamplingScreen.EveKeys = 0;
 SamplingScreen.EveProgressBarsCount = 0;
 SamplingScreen.EveProgressBars = 0;
 SamplingScreen.EveTogglesCount = 0;
 SamplingScreen.EveToggles = 0;
 SamplingScreen.EveButtonsCount = 8;
 SamplingScreen.EveButtons = SamplingScreen_EveButtons;
 SamplingScreen.EveTextsCount = 9;
 SamplingScreen.EveTexts = SamplingScreen_EveTexts;
 SamplingScreen.CEveTextsCount = 6;
 SamplingScreen.CEveTexts = SamplingScreen_CEveTexts;
 SamplingScreen.CEveNumbersCount = 0;
 SamplingScreen.CEveNumbers = 0;
```

```
SamplingScreen.DynResStart = 0;
SamplingScreen.Active = 0;
SamplingScreen.SniffObjectEvents = 0;
SamplingScreen.OnUp = 0;
SamplingScreen.OnDown = 0;
SamplingScreen.OnTagChange = 0;
SamplingScreen.OnPress = 0;


ProfilingScreen.Color = 0xFFFF;
ProfilingScreen.Width = 480;
ProfilingScreen.Height = 272;
ProfilingScreen.ObjectsCount = 76;
ProfilingScreen.BoxesCount = 1;
ProfilingScreen.Boxes = ProfilingScreen_Boxes;
ProfilingScreen.CBoxesCount = 2;
ProfilingScreen.CBoxes = ProfilingScreen_CBoxes;
ProfilingScreen.Boxes_RoundCount = 0;
ProfilingScreen.Boxes_Round = 0;
ProfilingScreen.CBoxes_RoundCount = 1;
ProfilingScreen.CBoxes_Round =
ProfilingScreen_CBoxes_Round;
ProfilingScreen.LinesCount = 50;
ProfilingScreen.Lines = ProfilingScreen_Lines;
ProfilingScreen.EveGaugesCount = 2;
ProfilingScreen.EveGauges = ProfilingScreen_EveGauges;
ProfilingScreen.EveKeysCount = 0;
ProfilingScreen.EveKeys = 0;
ProfilingScreen.EveProgressBarsCount = 0;
ProfilingScreen.EveProgressBars = 0;
ProfilingScreen.EveTogglesCount = 1;
ProfilingScreen.EveToggles = ProfilingScreen_EveToggles;
ProfilingScreen.EveButtonsCount = 3;
ProfilingScreen.EveButtons = ProfilingScreen_EveButtons;
ProfilingScreen.EveTextsCount = 3;
ProfilingScreen.EveTexts = ProfilingScreen_EveTexts;
ProfilingScreen.CEveTextsCount = 7;
ProfilingScreen.CEveTexts = ProfilingScreen_CEveTexts;
ProfilingScreen.CEveNumbersCount = 6;
ProfilingScreen.CEveNumbers = ProfilingScreen_CEveNumbers;
ProfilingScreen.DynResStart = 0;
ProfilingScreen.Active = 1;
ProfilingScreen.SniffObjectEvents = 0;
ProfilingScreen.OnUp = 0;
ProfilingScreen.OnDown = 0;
ProfilingScreen.OnTagChange = 0;
ProfilingScreen.OnPress = 0;


SettingScreen.Color = 0xFFFF;
SettingScreen.Width = 480;
SettingScreen.Height = 272;
SettingScreen.ObjectsCount = 13;
SettingScreen.BoxesCount = 0;
SettingScreen.Boxes = 0;
SettingScreen.CBoxesCount = 0;
SettingScreen.CBoxes = 0;
SettingScreen.Boxes_RoundCount = 0;
SettingScreen.Boxes_Round = 0;
SettingScreen.CBoxes_RoundCount = 2;
SettingScreen.CBoxes_Round = SettingScreen_CBoxes_Round;
SettingScreen.LinesCount = 0;
SettingScreen.Lines = 0;
SettingScreen.EveGaugesCount = 0;
SettingScreen.EveGauges = 0;
SettingScreen.EveKeysCount = 4;
SettingScreen.EveKeys = SettingScreen_EveKeys;
SettingScreen.EveProgressBarsCount = 0;
SettingScreen.EveProgressBars = 0;
SettingScreen.EveTogglesCount = 0;
SettingScreen.EveToggles = 0;
SettingScreen.EveButtonsCount = 6;
SettingScreen.EveButtons = SettingScreen_EveButtons;
SettingScreen.EveTextsCount = 0;
SettingScreen.EveTexts = 0;
SettingScreen.CEveTextsCount = 1;
SettingScreen.CEveTexts = SettingScreen_CEveTexts;
SettingScreen.CEveNumbersCount = 0;
SettingScreen.CEveNumbers = 0;
SettingScreen.DynResStart = 0;
SettingScreen.Active = 1;
SettingScreen.SniffObjectEvents = 1;
SettingScreen.OnUp = 0;
SettingScreen.OnDown = 0;
SettingScreen.OnTagChange = &SettingScreen_OnTagChange;
SettingScreen.OnPress = 0;

SettingScreen_OnTagChange.Action =
SettingScreenOnTagChange;
SettingScreen_OnTagChange.Sound.SndAct =
VTFT_SNDACT_NONE;
SettingScreen_OnTagChange.Sound.Effect = 0;
SettingScreen_OnTagChange.Sound.Pitch = 0;
SettingScreen_OnTagChange.Sound.Volume = 0;


SummaryScreen.Color = 0xFFFF;
SummaryScreen.Width = 480;
SummaryScreen.Height = 272;
SummaryScreen.ObjectsCount = 25;
SummaryScreen.BoxesCount = 0;
SummaryScreen.Boxes = 0;
SummaryScreen.CBoxesCount = 0;
SummaryScreen.CBoxes = 0;
SummaryScreen.Boxes_RoundCount = 1;
SummaryScreen.Boxes_Round = SummaryScreen_Boxes_Round;
SummaryScreen.CBoxes_RoundCount = 2;
SummaryScreen.CBoxes_Round =
SummaryScreen_CBoxes_Round;
SummaryScreen.LinesCount = 0;
SummaryScreen.Lines = 0;
SummaryScreen.EveGaugesCount = 0;
SummaryScreen.EveGauges = 0;
SummaryScreen.EveKeysCount = 0;
SummaryScreen.EveKeys = 0;
SummaryScreen.EveProgressBarsCount = 0;
SummaryScreen.EveProgressBars = 0;
SummaryScreen.EveTogglesCount = 0;
SummaryScreen.EveToggles = 0;
SummaryScreen.EveButtonsCount = 4;
SummaryScreen.EveButtons = SummaryScreen_EveButtons;
SummaryScreen.EveTextsCount = 13;
SummaryScreen.EveTexts = SummaryScreen_EveTexts;
SummaryScreen.CEveTextsCount = 5;
SummaryScreen.CEveTexts = SummaryScreen_CEveTexts;
SummaryScreen.CEveNumbersCount = 0;
SummaryScreen.CEveNumbers = 0;
SummaryScreen.DynResStart = 0;
SummaryScreen.Active = 0;
SummaryScreen.SniffObjectEvents = 0;
SummaryScreen.OnUp = 0;
SummaryScreen.OnDown = 0;
SummaryScreen.OnTagChange = 0;
SummaryScreen.OnPress = 0;

EveButtonSplashInit.OwnerScreen = &SplashScreen;
EveButtonSplashInit.Order = 1;
EveButtonSplashInit.Visible = 1;
EveButtonSplashInit.Opacity = 255;
EveButtonSplashInit.Tag = 255;
```

```
EveButtonSplashInit.Left = 150;
EveButtonSplashInit.Top = 185;
EveButtonSplashInit.Width = 170;
EveButtonSplashInit.Height = 69;
EveButtonSplashInit.Color = 0x0335;
EveButtonSplashInit.Press_Color = 0x7E3F;
EveButtonSplashInit.ColorTo = 0x7E3F;
EveButtonSplashInit.Press_ColorTo = 0x03DA;
EveButtonSplashInit.Caption = EveButtonSplashInit_Caption;
EveButtonSplashInit.FontName = 27;
EveButtonSplashInit.Font_Color = 0xFFFF;
EveButtonSplashInit.FontHandle = 27;
EveButtonSplashInit.Source = -1UL;
EveButtonSplashInit.Flat = 0;
EveButtonSplashInit.Active = 1;
EveButtonSplashInit.OnUp = 0;
EveButtonSplashInit.OnDown = 0;
EveButtonSplashInit.OnClick = &EveButtonSplashInit_OnClick;
EveButtonSplashInit.OnPress = 0;

EveButtonSplashInit_OnClick.Action =
EveButtonSplashInitOnClick;
EveButtonSplashInit_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
EveButtonSplashInit_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
EveButtonSplashInit_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
EveButtonSplashInit_OnClick.Sound.Volume = 255;

EveTextSplashScreenStatus.OwnerScreen = &SplashScreen;
EveTextSplashScreenStatus.Order = 2;
EveTextSplashScreenStatus.Visible = 0;
EveTextSplashScreenStatus.Opacity = 255;
EveTextSplashScreenStatus.Tag = 255;
EveTextSplashScreenStatus.Left = 196;
EveTextSplashScreenStatus.Top = 162;
EveTextSplashScreenStatus.Width = 68;
EveTextSplashScreenStatus.Height = 15;
EveTextSplashScreenStatus.Caption =
EveTextSplashScreenStatus_Caption;
EveTextSplashScreenStatus.TextAlign = taNone;
EveTextSplashScreenStatus.FontName = 26;
EveTextSplashScreenStatus.Font_Color = 0x0148;
EveTextSplashScreenStatus.FontHandle = 26;
EveTextSplashScreenStatus.Source = -1UL;
EveTextSplashScreenStatus.Active = 0;
EveTextSplashScreenStatus.OnUp = 0;
EveTextSplashScreenStatus.OnDown = 0;
EveTextSplashScreenStatus.OnClick = 0;
EveTextSplashScreenStatus.OnPress = 0;

EveProgressBar1.OwnerScreen = &SplashScreen;
EveProgressBar1.Order = 3;
EveProgressBar1.Visible = 0;
EveProgressBar1.Opacity = 255;
EveProgressBar1.Tag = 255;
EveProgressBar1.Left = 34;
EveProgressBar1.Top = 211;
EveProgressBar1.Width = 404;
EveProgressBar1.Height = 26;
EveProgressBar1.Background_Color = 0x0000;
EveProgressBar1.Color = 0x65BC;
EveProgressBar1.Value = 0;
EveProgressBar1.Range = 100;
EveProgressBar1.Flat = 0;

EveTextEncoderSamp.OwnerScreen = &SamplingScreen;
EveTextEncoderSamp.Order = 8;
EveTextEncoderSamp.Visible = 1;
EveTextEncoderSamp.Opacity = 255;
EveTextEncoderSamp.Tag = 255;
EveTextEncoderSamp.Left = 190;
EveTextEncoderSamp.Top = 12;
EveTextEncoderSamp.Width = 45;
EveTextEncoderSamp.Height = 17;
EveTextEncoderSamp.Caption = EveTextEncoderSamp_Caption;
EveTextEncoderSamp.TextAlign = taNone;
EveTextEncoderSamp.FontName = 27;
EveTextEncoderSamp.Font_Color = 0x0148;
EveTextEncoderSamp.FontHandle = 27;
EveTextEncoderSamp.Source = -1UL;
EveTextEncoderSamp.Active = 1;
EveTextEncoderSamp.OnUp = 0;
EveTextEncoderSamp.OnDown = 0;
EveTextEncoderSamp.OnClick = 0;
EveTextEncoderSamp.OnPress = 0;

EveTextLaserSmp.OwnerScreen = &SamplingScreen;
EveTextLaserSmp.Order = 9;
EveTextLaserSmp.Visible = 1;
EveTextLaserSmp.Opacity = 255;
EveTextLaserSmp.Tag = 255;
EveTextLaserSmp.Left = 190;
EveTextLaserSmp.Top = 36;
EveTextLaserSmp.Width = 45;
EveTextLaserSmp.Height = 17;
EveTextLaserSmp.Caption = EveTextLaserSmp_Caption;
EveTextLaserSmp.TextAlign = taNone;
EveTextLaserSmp.FontName = 27;
EveTextLaserSmp.Font_Color = 0x0148;
EveTextLaserSmp.FontHandle = 27;
EveTextLaserSmp.Source = -1UL;
EveTextLaserSmp.Active = 1;
EveTextLaserSmp.OnUp = 0;
EveTextLaserSmp.OnDown = 0;
EveTextLaserSmp.OnClick = 0;
EveTextLaserSmp.OnPress = 0;

EveTextPitchSmp.OwnerScreen = &SamplingScreen;
EveTextPitchSmp.Order = 10;
EveTextPitchSmp.Visible = 1;
EveTextPitchSmp.Opacity = 255;
EveTextPitchSmp.Tag = 255;
EveTextPitchSmp.Left = 190;
EveTextPitchSmp.Top = 64;
EveTextPitchSmp.Width = 45;
EveTextPitchSmp.Height = 17;
EveTextPitchSmp.Caption = EveTextPitchSmp_Caption;
EveTextPitchSmp.TextAlign = taNone;
EveTextPitchSmp.FontName = 27;
EveTextPitchSmp.Font_Color = 0x0148;
EveTextPitchSmp.FontHandle = 27;
EveTextPitchSmp.Source = -1UL;
EveTextPitchSmp.Active = 1;
EveTextPitchSmp.OnUp = 0;
EveTextPitchSmp.OnDown = 0;
EveTextPitchSmp.OnClick = 0;
EveTextPitchSmp.OnPress = 0;

EveTextGPSSmp.OwnerScreen = &SamplingScreen;
EveTextGPSSmp.Order = 11;
EveTextGPSSmp.Visible = 1;
EveTextGPSSmp.Opacity = 255;
EveTextGPSSmp.Tag = 255;
EveTextGPSSmp.Left = 190;
EveTextGPSSmp.Top = 117;
EveTextGPSSmp.Width = 62;
EveTextGPSSmp.Height = 17;
EveTextGPSSmp.Caption = EveTextGPSSmp_Caption;
```

```
EveTextGPSSmp.TextAlign = taNone;
EveTextGPSSmp.FontName = 27;
EveTextGPSSmp.Font_Color = 0x0148;
EveTextGPSSmp.FontHandle = 27;
EveTextGPSSmp.Source = -1UL;
EveTextGPSSmp.Active = 0;
EveTextGPSSmp.OnUp = 0;
EveTextGPSSmp.OnDown = 0;
EveTextGPSSmp.OnClick = 0;
EveTextGPSSmp.OnPress = 0;

EveTextSDSmp.OwnerScreen = &SamplingScreen;
EveTextSDSmp.Order = 12;
EveTextSDSmp.Visible = 1;
EveTextSDSmp.Opacity = 255;
EveTextSDSmp.Tag = 255;
EveTextSDSmp.Left = 190;
EveTextSDSmp.Top = 142;
EveTextSDSmp.Width = 45;
EveTextSDSmp.Height = 17;
EveTextSDSmp.Caption = EveTextSDSmp_Caption;
EveTextSDSmp.TextAlign = taNone;
EveTextSDSmp.FontName = 27;
EveTextSDSmp.Font_Color = 0x0148;
EveTextSDSmp.FontHandle = 27;
EveTextSDSmp.Source = -1UL;
EveTextSDSmp.Active = 1;
EveTextSDSmp.OnUp = 0;
EveTextSDSmp.OnDown = 0;
EveTextSDSmp.OnClick = 0;
EveTextSDSmp.OnPress = 0;

EveTextRollSmp.OwnerScreen = &SamplingScreen;
EveTextRollSmp.Order = 13;
EveTextRollSmp.Visible = 1;
EveTextRollSmp.Opacity = 255;
EveTextRollSmp.Tag = 255;
EveTextRollSmp.Left = 354;
EveTextRollSmp.Top = 66;
EveTextRollSmp.Width = 45;
EveTextRollSmp.Height = 17;
EveTextRollSmp.Caption = EveTextRollSmp_Caption;
EveTextRollSmp.TextAlign = taNone;
EveTextRollSmp.FontName = 27;
EveTextRollSmp.Font_Color = 0x0148;
EveTextRollSmp.FontHandle = 27;
EveTextRollSmp.Source = -1UL;
EveTextRollSmp.Active = 1;
EveTextRollSmp.OnUp = 0;
EveTextRollSmp.OnDown = 0;
EveTextRollSmp.OnClick = 0;
EveTextRollSmp.OnPress = 0;

EveTextAccx.OwnerScreen = &SamplingScreen;
EveTextAccx.Order = 14;
EveTextAccx.Visible = 1;
EveTextAccx.Opacity = 255;
EveTextAccx.Tag = 255;
EveTextAccx.Left = 190;
EveTextAccx.Top = 91;
EveTextAccx.Width = 45;
EveTextAccx.Height = 17;
EveTextAccx.Caption = EveTextAccx_Caption;
EveTextAccx.TextAlign = taNone;
EveTextAccx.FontName = 27;
EveTextAccx.Font_Color = 0x0148;
EveTextAccx.FontHandle = 27;
EveTextAccx.Source = -1UL;
EveTextAccx.Active = 1;
EveTextAccx.OnUp = 0;
EveTextAccx.OnDown = 0;
EveTextAccx.OnClick = 0;
EveTextAccx.OnPress = 0;

EveTextAccy.OwnerScreen = &SamplingScreen;
EveTextAccy.Order = 15;
EveTextAccy.Visible = 1;
EveTextAccy.Opacity = 255;
EveTextAccy.Tag = 255;
EveTextAccy.Left = 306;
EveTextAccy.Top = 93;
EveTextAccy.Width = 45;
EveTextAccy.Height = 17;
EveTextAccy.Caption = EveTextAccy_Caption;
EveTextAccy.TextAlign = taNone;
EveTextAccy.FontName = 27;
EveTextAccy.Font_Color = 0x0148;
EveTextAccy.FontHandle = 27;
EveTextAccy.Source = -1UL;
EveTextAccy.Active = 1;
EveTextAccy.OnUp = 0;
EveTextAccy.OnDown = 0;
EveTextAccy.OnClick = 0;
EveTextAccy.OnPress = 0;

EveTextAccz.OwnerScreen = &SamplingScreen;
EveTextAccz.Order = 16;
EveTextAccz.Visible = 1;
EveTextAccz.Opacity = 255;
EveTextAccz.Tag = 255;
EveTextAccz.Left = 401;
EveTextAccz.Top = 94;
EveTextAccz.Width = 45;
EveTextAccz.Height = 17;
EveTextAccz.Caption = EveTextAccz_Caption;
EveTextAccz.TextAlign = taNone;
EveTextAccz.FontName = 27;
EveTextAccz.Font_Color = 0x0148;
EveTextAccz.FontHandle = 27;
EveTextAccz.Source = -1UL;
EveTextAccz.Active = 1;
EveTextAccz.OnUp = 0;
EveTextAccz.OnDown = 0;
EveTextAccz.OnClick = 0;
EveTextAccz.OnPress = 0;

BoxRound7.OwnerScreen = &SamplingScreen;
BoxRound7.Order = 17;
BoxRound7.Visible = 0;
BoxRound7.Opacity = 255;
BoxRound7.Tag = 255;
BoxRound7.Left = 6;
BoxRound7.Top = 187;
BoxRound7.Width = 470;
BoxRound7.Height = 81;
BoxRound7.Pen_Width = 1;
BoxRound7.Pen_Color = 0x0000;
BoxRound7.Color = 0xC618;
BoxRound7.Press_Color = 0xE71C;
BoxRound7.Corner_Radius = 3;
BoxRound7.Active = 0;
BoxRound7.OnUp = 0;
BoxRound7.OnDown = 0;
BoxRound7.OnClick = 0;
BoxRound7.OnPress = 0;

EveButtonsmpJogRear.OwnerScreen = &SamplingScreen;
EveButtonsmpJogRear.Order = 18;
EveButtonsmpJogRear.Visible = 0;
EveButtonsmpJogRear.Opacity = 255;
```

```
 EveButtonsmpJogRear.Tag = 255;
 EveButtonsmpJogRear.Left = 16;
 EveButtonsmpJogRear.Top = 192;
 EveButtonsmpJogRear.Width = 121;
 EveButtonsmpJogRear.Height = 69;
 EveButtonsmpJogRear.Color = 0x03DA;
 EveButtonsmpJogRear.Press_Color = 0x7E3F;
 EveButtonsmpJogRear.ColorTo = 0x7E3F;
 EveButtonsmpJogRear.Press_ColorTo = 0x03DA;
 EveButtonsmpJogRear.Caption =
EveButtonsmpJogRear_Caption;
 EveButtonsmpJogRear.FontName = 27;
 EveButtonsmpJogRear.Font_Color = 0xFFFF;
 EveButtonsmpJogRear.FontHandle = 27;
 EveButtonsmpJogRear.Source = -1UL;
 EveButtonsmpJogRear.Flat = 0;
 EveButtonsmpJogRear.Active = 0;
 EveButtonsmpJogRear.OnUp = 0;
 EveButtonsmpJogRear.OnDown = 0;
 EveButtonsmpJogRear.OnClick = 0;
 EveButtonsmpJogRear.OnPress =
&EveButtonsmpJogRear_OnPress;

 EveButtonsmpJogRear_OnPress.Action =
EveButtonsmpJogRearOnPress;
 EveButtonsmpJogRear_OnPress.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonsmpJogRear_OnPress.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonsmpJogRear_OnPress.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonsmpJogRear_OnPress.Sound.Volume = 255;

 EveButtonsmpJogFront.OwnerScreen = &SamplingScreen;
 EveButtonsmpJogFront.Order = 19;
 EveButtonsmpJogFront.Visible = 0;
 EveButtonsmpJogFront.Opacity = 255;
 EveButtonsmpJogFront.Tag = 255;
 EveButtonsmpJogFront.Left = 143;
 EveButtonsmpJogFront.Top = 192;
 EveButtonsmpJogFront.Width = 136;
 EveButtonsmpJogFront.Height = 69;
 EveButtonsmpJogFront.Color = 0x03DA;
 EveButtonsmpJogFront.Press_Color = 0x7E3F;
 EveButtonsmpJogFront.ColorTo = 0x7E3F;
 EveButtonsmpJogFront.Press_ColorTo = 0x03DA;
 EveButtonsmpJogFront.Caption =
EveButtonsmpJogFront_Caption;
 EveButtonsmpJogFront.FontName = 27;
 EveButtonsmpJogFront.Font_Color = 0xFFFF;
 EveButtonsmpJogFront.FontHandle = 27;
 EveButtonsmpJogFront.Source = -1UL;
 EveButtonsmpJogFront.Flat = 0;
 EveButtonsmpJogFront.Active = 0;
 EveButtonsmpJogFront.OnUp = 0;
 EveButtonsmpJogFront.OnDown = 0;
 EveButtonsmpJogFront.OnClick = 0;
 EveButtonsmpJogFront.OnPress =
&EveButtonsmpJogFront_OnPress;

 EveButtonsmpJogFront_OnPress.Action =
EveButtonsmpJogFrontOnPress;
 EveButtonsmpJogFront_OnPress.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonsmpJogFront_OnPress.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonsmpJogFront_OnPress.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonsmpJogFront_OnPress.Sound.Volume = 255;

 EveButtonsmpBack.OwnerScreen = &SamplingScreen;
 EveButtonsmpBack.Order = 20;
 EveButtonsmpBack.Visible = 0;
 EveButtonsmpBack.Opacity = 255;
 EveButtonsmpBack.Tag = 255;
 EveButtonsmpBack.Left = 308;
 EveButtonsmpBack.Top = 192;
 EveButtonsmpBack.Width = 163;
 EveButtonsmpBack.Height = 69;
 EveButtonsmpBack.Color = 0x9824;
 EveButtonsmpBack.Press_Color = 0x7E3F;
 EveButtonsmpBack.ColorTo = 0xF434;
 EveButtonsmpBack.Press_ColorTo = 0x03DA;
 EveButtonsmpBack.Caption = EveButtonsmpBack_Caption;
 EveButtonsmpBack.FontName = 27;
 EveButtonsmpBack.Font_Color = 0xFFFF;
 EveButtonsmpBack.FontHandle = 27;
 EveButtonsmpBack.Source = -1UL;
 EveButtonsmpBack.Flat = 0;
 EveButtonsmpBack.Active = 0;
 EveButtonsmpBack.OnUp = 0;
 EveButtonsmpBack.OnDown = 0;
 EveButtonsmpBack.OnClick = &EveButtonsmpBack_OnClick;
 EveButtonsmpBack.OnPress = 0;

 EveButtonsmpBack_OnClick.Action =
EveButtonsmpBackOnClick;
 EveButtonsmpBack_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonsmpBack_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonsmpBack_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonsmpBack_OnClick.Sound.Volume = 255;

 BoxRound8.OwnerScreen = &SamplingScreen;
 BoxRound8.Order = 21;
 BoxRound8.Visible = 1;
 BoxRound8.Opacity = 255;
 BoxRound8.Tag = 255;
 BoxRound8.Left = 6;
 BoxRound8.Top = 187;
 BoxRound8.Width = 470;
 BoxRound8.Height = 81;
 BoxRound8.Pen_Width = 1;
 BoxRound8.Pen_Color = 0x0000;
 BoxRound8.Color = 0xC618;
 BoxRound8.Press_Color = 0xE71C;
 BoxRound8.Corner_Radius = 3;
 BoxRound8.Active = 0;
 BoxRound8.OnUp = 0;
 BoxRound8.OnDown = 0;
 BoxRound8.OnClick = 0;
 BoxRound8.OnPress = 0;

 EveButtonSmpStartSampling.OwnerScreen = &SamplingScreen;
 EveButtonSmpStartSampling.Order = 22;
 EveButtonSmpStartSampling.Visible = 1;
 EveButtonSmpStartSampling.Opacity = 255;
 EveButtonSmpStartSampling.Tag = 255;
 EveButtonSmpStartSampling.Left = 16;
 EveButtonSmpStartSampling.Top = 192;
 EveButtonSmpStartSampling.Width = 121;
 EveButtonSmpStartSampling.Height = 69;
 EveButtonSmpStartSampling.Color = 0x03DA;
 EveButtonSmpStartSampling.Press_Color = 0x7E3F;
 EveButtonSmpStartSampling.ColorTo = 0x7E3F;
 EveButtonSmpStartSampling.Press_ColorTo = 0x03DA;
 EveButtonSmpStartSampling.Caption =
EveButtonSmpStartSampling_Caption;
```

```
 EveButtonSmpStartSampling.FontName = 27;
 EveButtonSmpStartSampling.Font_Color = 0xFFFF;
 EveButtonSmpStartSampling.FontHandle = 27;
 EveButtonSmpStartSampling.Source = -1UL;
 EveButtonSmpStartSampling.Flat = 0;
 EveButtonSmpStartSampling.Active = 1;
 EveButtonSmpStartSampling.OnUp = 0;
 EveButtonSmpStartSampling.OnDown = 0;
 EveButtonSmpStartSampling.OnClick =
&EveButtonSmpStartSampling_OnClick;
 EveButtonSmpStartSampling.OnPress = 0;

 EveButtonSmpStartSampling_OnClick.Action =
EveButtonSmpStartSamplingOnClick;
 EveButtonSmpStartSampling_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonSmpStartSampling_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonSmpStartSampling_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonSmpStartSampling_OnClick.Sound.Volume = 255;

 EveButtonSmpStopSampling.OwnerScreen = &SamplingScreen;
 EveButtonSmpStopSampling.Order = 23;
 EveButtonSmpStopSampling.Visible = 0;
 EveButtonSmpStopSampling.Opacity = 255;
 EveButtonSmpStopSampling.Tag = 255;
 EveButtonSmpStopSampling.Left = 16;
 EveButtonSmpStopSampling.Top = 192;
 EveButtonSmpStopSampling.Width = 121;
 EveButtonSmpStopSampling.Height = 69;
 EveButtonSmpStopSampling.Color = 0x03DA;
 EveButtonSmpStopSampling.Press_Color = 0x7E3F;
 EveButtonSmpStopSampling.ColorTo = 0x7E3F;
 EveButtonSmpStopSampling.Press_ColorTo = 0x03DA;
 EveButtonSmpStopSampling.Caption =
EveButtonSmpStopSampling_Caption;
 EveButtonSmpStopSampling.FontName = 27;
 EveButtonSmpStopSampling.Font_Color = 0xFFFF;
 EveButtonSmpStopSampling.FontHandle = 27;
 EveButtonSmpStopSampling.Source = -1UL;
 EveButtonSmpStopSampling.Flat = 0;
 EveButtonSmpStopSampling.Active = 0;
 EveButtonSmpStopSampling.OnUp = 0;
 EveButtonSmpStopSampling.OnDown = 0;
 EveButtonSmpStopSampling.OnClick =
&EveButtonSmpStopSampling_OnClick;
 EveButtonSmpStopSampling.OnPress = 0;

 EveButtonSmpStopSampling_OnClick.Action =
EveButtonSmpStopSamplingOnClick;
 EveButtonSmpStopSampling_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonSmpStopSampling_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonSmpStopSampling_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonSmpStopSampling_OnClick.Sound.Volume = 255;

 EveButtonSmpConfigureMotor.OwnerScreen =
&SamplingScreen;
 EveButtonSmpConfigureMotor.Order = 24;
 EveButtonSmpConfigureMotor.Visible = 1;
 EveButtonSmpConfigureMotor.Opacity = 255;
 EveButtonSmpConfigureMotor.Tag = 255;
 EveButtonSmpConfigureMotor.Left = 143;
 EveButtonSmpConfigureMotor.Top = 192;
 EveButtonSmpConfigureMotor.Width = 136;
 EveButtonSmpConfigureMotor.Height = 69;
 EveButtonSmpConfigureMotor.Color = 0xA865;
 EveButtonSmpConfigureMotor.Press_Color = 0x7E3F;
 EveButtonSmpConfigureMotor.ColorTo = 0xF3B5;
 EveButtonSmpConfigureMotor.Press_ColorTo = 0x03DA;
 EveButtonSmpConfigureMotor.Caption =
EveButtonSmpConfigureMotor_Caption;
 EveButtonSmpConfigureMotor.FontName = 27;
 EveButtonSmpConfigureMotor.Font_Color = 0xFFFF;
 EveButtonSmpConfigureMotor.FontHandle = 27;
 EveButtonSmpConfigureMotor.Source = -1UL;
 EveButtonSmpConfigureMotor.Flat = 0;
 EveButtonSmpConfigureMotor.Active = 1;
 EveButtonSmpConfigureMotor.OnUp = 0;
 EveButtonSmpConfigureMotor.OnDown = 0;
 EveButtonSmpConfigureMotor.OnClick =
&EveButtonSmpConfigureMotor_OnClick;
 EveButtonSmpConfigureMotor.OnPress = 0;

 EveButtonSmpConfigureMotor_OnClick.Action =
EveButtonSmpConfigureMotorOnClick;
 EveButtonSmpConfigureMotor_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonSmpConfigureMotor_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonSmpConfigureMotor_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonSmpConfigureMotor_OnClick.Sound.Volume = 255;

 EveButtonSmpGotoProf.OwnerScreen = &SamplingScreen;
 EveButtonSmpGotoProf.Order = 25;
 EveButtonSmpGotoProf.Visible = 1;
 EveButtonSmpGotoProf.Opacity = 255;
 EveButtonSmpGotoProf.Tag = 255;
 EveButtonSmpGotoProf.Left = 308;
 EveButtonSmpGotoProf.Top = 192;
 EveButtonSmpGotoProf.Width = 163;
 EveButtonSmpGotoProf.Height = 69;
 EveButtonSmpGotoProf.Color = 0x14A7;
 EveButtonSmpGotoProf.Press_Color = 0x7E3F;
 EveButtonSmpGotoProf.ColorTo = 0x8FB7;
 EveButtonSmpGotoProf.Press_ColorTo = 0x03DA;
 EveButtonSmpGotoProf.Caption =
EveButtonSmpGotoProf_Caption;
 EveButtonSmpGotoProf.FontName = 27;
 EveButtonSmpGotoProf.Font_Color = 0xFFFF;
 EveButtonSmpGotoProf.FontHandle = 27;
 EveButtonSmpGotoProf.Source = -1UL;
 EveButtonSmpGotoProf.Flat = 0;
 EveButtonSmpGotoProf.Active = 1;
 EveButtonSmpGotoProf.OnUp = 0;
 EveButtonSmpGotoProf.OnDown = 0;
 EveButtonSmpGotoProf.OnClick =
&EveButtonSmpGotoProf_OnClick;
 EveButtonSmpGotoProf.OnPress = 0;

 EveButtonSmpGotoProf_OnClick.Action =
EveButtonSmpGotoProfOnClick;
 EveButtonSmpGotoProf_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonSmpGotoProf_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonSmpGotoProf_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonSmpGotoProf_OnClick.Sound.Volume = 255;

 EveButtonsmpResetInc.OwnerScreen = &SamplingScreen;
 EveButtonsmpResetInc.Order = 26;
 EveButtonsmpResetInc.Visible = 0;
 EveButtonsmpResetInc.Opacity = 255;
 EveButtonsmpResetInc.Tag = 255;
 EveButtonsmpResetInc.Left = 332;
```

```
 EveButtonsmpResetInc.Top = 8;
 EveButtonsmpResetInc.Width = 136;
 EveButtonsmpResetInc.Height = 53;
 EveButtonsmpResetInc.Color = 0x03DA;
 EveButtonsmpResetInc.Press_Color = 0x7E3F;
 EveButtonsmpResetInc.ColorTo = 0x7E3F;
 EveButtonsmpResetInc.Press_ColorTo = 0x03DA;
 EveButtonsmpResetInc.Caption =
EveButtonsmpResetInc_Caption;
 EveButtonsmpResetInc.FontName = 27;
 EveButtonsmpResetInc.Font_Color = 0xFFFF;
 EveButtonsmpResetInc.FontHandle = 27;
 EveButtonsmpResetInc.Source = -1UL;
 EveButtonsmpResetInc.Flat = 0;
 EveButtonsmpResetInc.Active = 0;
 EveButtonsmpResetInc.OnUp = 0;
 EveButtonsmpResetInc.OnDown = 0;
 EveButtonsmpResetInc.OnClick =
&EveButtonsmpResetInc_OnClick;
 EveButtonsmpResetInc.OnPress = 0;

 EveButtonsmpResetInc_OnClick.Action =
EveButtonsmpResetIncOnClick;
 EveButtonsmpResetInc_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonsmpResetInc_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonsmpResetInc_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonsmpResetInc_OnClick.Sound.Volume = 255;

 EveGaugeProfSpeed.OwnerScreen = &ProfilingScreen;
 EveGaugeProfSpeed.Order = 11;
 EveGaugeProfSpeed.Visible = 1;
 EveGaugeProfSpeed.Opacity = 255;
 EveGaugeProfSpeed.Tag = 255;
 EveGaugeProfSpeed.Left = 280;
 EveGaugeProfSpeed.Top = 45;
 EveGaugeProfSpeed.Radius = 40;
 EveGaugeProfSpeed.Pen_Width = 1;
 EveGaugeProfSpeed.Pen_Color = 0x0000;
 EveGaugeProfSpeed.Color = 0x03DA;
 EveGaugeProfSpeed.Press_Color = 0x7E3F;
 EveGaugeProfSpeed.Major = 10;
 EveGaugeProfSpeed.Minor = 5;
 EveGaugeProfSpeed.Value = 0;
 EveGaugeProfSpeed.Range = 100;
 EveGaugeProfSpeed.Flat = 0;
 EveGaugeProfSpeed.NoBackground = 0;
 EveGaugeProfSpeed.NoPointer = 0;
 EveGaugeProfSpeed.TicksVisible = 1;
 EveGaugeProfSpeed.Active = 0;
 EveGaugeProfSpeed.OnUp = 0;
 EveGaugeProfSpeed.OnDown = 0;
 EveGaugeProfSpeed.OnClick = 0;
 EveGaugeProfSpeed.OnPress = 0;

 EveGaugeProfCrossSlp.OwnerScreen = &ProfilingScreen;
 EveGaugeProfCrossSlp.Order = 12;
 EveGaugeProfCrossSlp.Visible = 1;
 EveGaugeProfCrossSlp.Opacity = 255;
 EveGaugeProfCrossSlp.Tag = 255;
 EveGaugeProfCrossSlp.Left = 386;
 EveGaugeProfCrossSlp.Top = 45;
 EveGaugeProfCrossSlp.Radius = 40;
 EveGaugeProfCrossSlp.Pen_Width = 1;
 EveGaugeProfCrossSlp.Pen_Color = 0x0000;
 EveGaugeProfCrossSlp.Color = 0x03DA;
 EveGaugeProfCrossSlp.Press_Color = 0x7E3F;
 EveGaugeProfCrossSlp.Major = 10;
 EveGaugeProfCrossSlp.Minor = 5;
 EveGaugeProfCrossSlp.Value = 50;
 EveGaugeProfCrossSlp.Range = 100;
 EveGaugeProfCrossSlp.Flat = 0;
 EveGaugeProfCrossSlp.NoBackground = 0;
 EveGaugeProfCrossSlp.NoPointer = 0;
 EveGaugeProfCrossSlp.TicksVisible = 1;
 EveGaugeProfCrossSlp.Active = 0;
 EveGaugeProfCrossSlp.OnUp = 0;
 EveGaugeProfCrossSlp.OnDown = 0;
 EveGaugeProfCrossSlp.OnClick = 0;
 EveGaugeProfCrossSlp.OnPress = 0;

 EveToggleProfMode.OwnerScreen = &ProfilingScreen;
 EveToggleProfMode.Order = 15;
 EveToggleProfMode.Visible = 1;
 EveToggleProfMode.Opacity = 255;
 EveToggleProfMode.Tag = 255;
 EveToggleProfMode.Left = 335;
 EveToggleProfMode.Top = 134;
 EveToggleProfMode.Width = 76;
 EveToggleProfMode.Height = 18;
 EveToggleProfMode.Pen_Width = 1;
 EveToggleProfMode.Pen_Color = 0x0000;
 EveToggleProfMode.Background_Color = 0x03DA;
 EveToggleProfMode.Color = 0xFFFF;
 EveToggleProfMode.Press_Color = 0x7E3F;
 EveToggleProfMode.StateOFF_Caption =
EveToggleProfMode_StateOFF_Caption;
 EveToggleProfMode.StateON_Caption =
EveToggleProfMode_StateON_Caption;
 EveToggleProfMode.FontName = 26;
 EveToggleProfMode.Font_Color = 0xFFFF;
 EveToggleProfMode.FontHandle = 26;
 EveToggleProfMode.Source = -1UL;
 EveToggleProfMode.State = 0;
 EveToggleProfMode.Flat = 0;
 EveToggleProfMode.Active = 1;
 EveToggleProfMode.OnUp = 0;
 EveToggleProfMode.OnDown = 0;
 EveToggleProfMode.OnClick = &EveToggleProfMode_OnClick;
 EveToggleProfMode.OnPress = 0;

 EveToggleProfMode_OnClick.Action =
EveToggleProfModeOnClick;
 EveToggleProfMode_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveToggleProfMode_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveToggleProfMode_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveToggleProfMode_OnClick.Sound.Volume = 255;

 EveButtonProfStartProfiling.OwnerScreen = &ProfilingScreen;
 EveButtonProfStartProfiling.Order = 17;
 EveButtonProfStartProfiling.Visible = 1;
 EveButtonProfStartProfiling.Opacity = 255;
 EveButtonProfStartProfiling.Tag = 255;
 EveButtonProfStartProfiling.Left = 12;
 EveButtonProfStartProfiling.Top = 220;
 EveButtonProfStartProfiling.Width = 148;
 EveButtonProfStartProfiling.Height = 48;
 EveButtonProfStartProfiling.Color = 0x0C20;
 EveButtonProfStartProfiling.Press_Color = 0x7E3F;
 EveButtonProfStartProfiling.ColorTo = 0x7757;
 EveButtonProfStartProfiling.Press_ColorTo = 0x03DA;
 EveButtonProfStartProfiling.Caption =
EveButtonProfStartProfiling_Caption;
 EveButtonProfStartProfiling.FontName = 27;
 EveButtonProfStartProfiling.Font_Color = 0xFFFF;
```

```
 EveButtonProfStartProfiling.FontHandle = 27;
 EveButtonProfStartProfiling.Source = -1UL;
 EveButtonProfStartProfiling.Flat = 0;
 EveButtonProfStartProfiling.Active = 1;
 EveButtonProfStartProfiling.OnUp = 0;
 EveButtonProfStartProfiling.OnDown = 0;
 EveButtonProfStartProfiling.OnClick =
&EveButtonProfStartProfiling_OnClick;
 EveButtonProfStartProfiling.OnPress = 0;

 EveButtonProfStartProfiling_OnClick.Action =
EveButtonProfStartProfilingOnClick;
 EveButtonProfStartProfiling_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonProfStartProfiling_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonProfStartProfiling_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonProfStartProfiling_OnClick.Sound.Volume = 255;

 EveButtonProfStopProfiling.OwnerScreen = &ProfilingScreen;
 EveButtonProfStopProfiling.Order = 18;
 EveButtonProfStopProfiling.Visible = 0;
 EveButtonProfStopProfiling.Opacity = 255;
 EveButtonProfStopProfiling.Tag = 255;
 EveButtonProfStopProfiling.Left = 12;
 EveButtonProfStopProfiling.Top = 220;
 EveButtonProfStopProfiling.Width = 148;
 EveButtonProfStopProfiling.Height = 48;
 EveButtonProfStopProfiling.Color = 0xC000;
 EveButtonProfStopProfiling.Press_Color = 0x7E3F;
 EveButtonProfStopProfiling.ColorTo = 0xEB73;
 EveButtonProfStopProfiling.Press_ColorTo = 0x03DA;
 EveButtonProfStopProfiling.Caption =
EveButtonProfStopProfiling_Caption;
 EveButtonProfStopProfiling.FontName = 27;
 EveButtonProfStopProfiling.Font_Color = 0xFFFF;
 EveButtonProfStopProfiling.FontHandle = 27;
 EveButtonProfStopProfiling.Source = -1UL;
 EveButtonProfStopProfiling.Flat = 0;
 EveButtonProfStopProfiling.Active = 0;
 EveButtonProfStopProfiling.OnUp = 0;
 EveButtonProfStopProfiling.OnDown = 0;
 EveButtonProfStopProfiling.OnClick =
&EveButtonProfStopProfiling_OnClick;
 EveButtonProfStopProfiling.OnPress = 0;

 EveButtonProfStopProfiling_OnClick.Action =
EveButtonProfStopProfilingOnClick;
 EveButtonProfStopProfiling_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonProfStopProfiling_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonProfStopProfiling_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonProfStopProfiling_OnClick.Sound.Volume = 255;

 Line1.OwnerScreen = &ProfilingScreen;
 Line1.Order = 19;
 Line1.Visible = 0;
 Line1.Opacity = 255;
 Line1.Tag = 255;
 Line1.First_Point_X = 40;
 Line1.First_Point_Y = 32;
 Line1.Second_Point_X = 50;
 Line1.Second_Point_Y = 32;
 Line1.Pen_Width = 3;
 Line1.Pen_Color = 0x1515;

 Line2.OwnerScreen = &ProfilingScreen;
 Line2.Order = 20;
 Line2.Visible = 0;
 Line2.Opacity = 255;
 Line2.Tag = 255;
 Line2.First_Point_X = 40;
 Line2.First_Point_Y = 32;
 Line2.Second_Point_X = 50;
 Line2.Second_Point_Y = 32;
 Line2.Pen_Width = 3;
 Line2.Pen_Color = 0x1515;

 Line3.OwnerScreen = &ProfilingScreen;
 Line3.Order = 21;
 Line3.Visible = 0;
 Line3.Opacity = 255;
 Line3.Tag = 255;
 Line3.First_Point_X = 40;
 Line3.First_Point_Y = 32;
 Line3.Second_Point_X = 50;
 Line3.Second_Point_Y = 32;
 Line3.Pen_Width = 3;
 Line3.Pen_Color = 0x1515;

 Line4.OwnerScreen = &ProfilingScreen;
 Line4.Order = 22;
 Line4.Visible = 0;
 Line4.Opacity = 255;
 Line4.Tag = 255;
 Line4.First_Point_X = 40;
 Line4.First_Point_Y = 32;
 Line4.Second_Point_X = 50;
 Line4.Second_Point_Y = 32;
 Line4.Pen_Width = 3;
 Line4.Pen_Color = 0x1515;

 Line5.OwnerScreen = &ProfilingScreen;
 Line5.Order = 23;
 Line5.Visible = 0;
 Line5.Opacity = 255;
 Line5.Tag = 255;
 Line5.First_Point_X = 40;
 Line5.First_Point_Y = 32;
 Line5.Second_Point_X = 50;
 Line5.Second_Point_Y = 32;
 Line5.Pen_Width = 3;
 Line5.Pen_Color = 0x1515;

 Line6.OwnerScreen = &ProfilingScreen;
 Line6.Order = 24;
 Line6.Visible = 0;
 Line6.Opacity = 255;
 Line6.Tag = 255;
 Line6.First_Point_X = 40;
 Line6.First_Point_Y = 32;
 Line6.Second_Point_X = 50;
 Line6.Second_Point_Y = 32;
 Line6.Pen_Width = 3;
 Line6.Pen_Color = 0x1515;

 Line7.OwnerScreen = &ProfilingScreen;
 Line7.Order = 25;
 Line7.Visible = 0;
 Line7.Opacity = 255;
 Line7.Tag = 255;
 Line7.First_Point_X = 40;
 Line7.First_Point_Y = 32;
 Line7.Second_Point_X = 50;
 Line7.Second_Point_Y = 32;
 Line7.Pen_Width = 3;
 Line7.Pen_Color = 0x1515;
```

```
Line8.OwnerScreen = &ProfilingScreen;
Line8.Order = 26;
Line8.Visible = 0;
Line8.Opacity = 255;
Line8.Tag = 255;
Line8.First_Point_X = 40;
Line8.First_Point_Y = 32;
Line8.Second_Point_X = 50;
Line8.Second_Point_Y = 32;
Line8.Pen_Width = 3;
Line8.Pen_Color = 0x1515;

Line9.OwnerScreen = &ProfilingScreen;
Line9.Order = 27;
Line9.Visible = 0;
Line9.Opacity = 255;
Line9.Tag = 255;
Line9.First_Point_X = 40;
Line9.First_Point_Y = 32;
Line9.Second_Point_X = 50;
Line9.Second_Point_Y = 32;
Line9.Pen_Width = 3;
Line9.Pen_Color = 0x1515;

Line10.OwnerScreen = &ProfilingScreen;
Line10.Order = 28;
Line10.Visible = 0;
Line10.Opacity = 255;
Line10.Tag = 255;
Line10.First_Point_X = 40;
Line10.First_Point_Y = 32;
Line10.Second_Point_X = 50;
Line10.Second_Point_Y = 32;
Line10.Pen_Width = 3;
Line10.Pen_Color = 0x1515;

Line11.OwnerScreen = &ProfilingScreen;
Line11.Order = 29;
Line11.Visible = 0;
Line11.Opacity = 255;
Line11.Tag = 255;
Line11.First_Point_X = 40;
Line11.First_Point_Y = 32;
Line11.Second_Point_X = 50;
Line11.Second_Point_Y = 32;
Line11.Pen_Width = 3;
Line11.Pen_Color = 0x1515;

Line12.OwnerScreen = &ProfilingScreen;
Line12.Order = 30;
Line12.Visible = 0;
Line12.Opacity = 255;
Line12.Tag = 255;
Line12.First_Point_X = 40;
Line12.First_Point_Y = 32;
Line12.Second_Point_X = 50;
Line12.Second_Point_Y = 32;
Line12.Pen_Width = 3;
Line12.Pen_Color = 0x1515;

Line13.OwnerScreen = &ProfilingScreen;
Line13.Order = 31;
Line13.Visible = 0;
Line13.Opacity = 255;
Line13.Tag = 255;
Line13.First_Point_X = 40;
Line13.First_Point_Y = 32;
Line13.Second_Point_X = 50;
Line13.Second_Point_Y = 32;
Line13.Pen_Width = 3;
Line13.Pen_Color = 0x1515;

Line14.OwnerScreen = &ProfilingScreen;
Line14.Order = 32;
Line14.Visible = 0;
Line14.Opacity = 255;
Line14.Tag = 255;
Line14.First_Point_X = 40;
Line14.First_Point_Y = 32;
Line14.Second_Point_X = 50;
Line14.Second_Point_Y = 32;
Line14.Pen_Width = 3;
Line14.Pen_Color = 0x1515;

Line15.OwnerScreen = &ProfilingScreen;
Line15.Order = 33;
Line15.Visible = 0;
Line15.Opacity = 255;
Line15.Tag = 255;
Line15.First_Point_X = 40;
Line15.First_Point_Y = 32;
Line15.Second_Point_X = 50;
Line15.Second_Point_Y = 32;
Line15.Pen_Width = 3;
Line15.Pen_Color = 0x1515;

Line16.OwnerScreen = &ProfilingScreen;
Line16.Order = 34;
Line16.Visible = 0;
Line16.Opacity = 255;
Line16.Tag = 255;
Line16.First_Point_X = 40;
Line16.First_Point_Y = 32;
Line16.Second_Point_X = 50;
Line16.Second_Point_Y = 32;
Line16.Pen_Width = 3;
Line16.Pen_Color = 0x1515;

Line17.OwnerScreen = &ProfilingScreen;
Line17.Order = 35;
Line17.Visible = 0;
Line17.Opacity = 255;
Line17.Tag = 255;
Line17.First_Point_X = 40;
Line17.First_Point_Y = 32;
Line17.Second_Point_X = 50;
Line17.Second_Point_Y = 32;
Line17.Pen_Width = 3;
Line17.Pen_Color = 0x1515;

Line18.OwnerScreen = &ProfilingScreen;
Line18.Order = 36;
Line18.Visible = 0;
Line18.Opacity = 255;
Line18.Tag = 255;
Line18.First_Point_X = 40;
Line18.First_Point_Y = 32;
Line18.Second_Point_X = 50;
Line18.Second_Point_Y = 32;
Line18.Pen_Width = 3;
Line18.Pen_Color = 0x1515;

Line19.OwnerScreen = &ProfilingScreen;
Line19.Order = 37;
Line19.Visible = 0;
Line19.Opacity = 255;
Line19.Tag = 255;
Line19.First_Point_X = 40;
Line19.First_Point_Y = 32;
```

```
Line19.Second_Point_X = 50;
Line19.Second_Point_Y = 32;
Line19.Pen_Width = 3;
Line19.Pen_Color = 0x1515;

Line20.OwnerScreen = &ProfilingScreen;
Line20.Order = 38;
Line20.Visible = 0;
Line20.Opacity = 255;
Line20.Tag = 255;
Line20.First_Point_X = 40;
Line20.First_Point_Y = 32;
Line20.Second_Point_X = 50;
Line20.Second_Point_Y = 32;
Line20.Pen_Width = 3;
Line20.Pen_Color = 0x1515;

Line21.OwnerScreen = &ProfilingScreen;
Line21.Order = 39;
Line21.Visible = 0;
Line21.Opacity = 255;
Line21.Tag = 255;
Line21.First_Point_X = 40;
Line21.First_Point_Y = 32;
Line21.Second_Point_X = 50;
Line21.Second_Point_Y = 32;
Line21.Pen_Width = 3;
Line21.Pen_Color = 0x1515;

Line22.OwnerScreen = &ProfilingScreen;
Line22.Order = 40;
Line22.Visible = 0;
Line22.Opacity = 255;
Line22.Tag = 255;
Line22.First_Point_X = 40;
Line22.First_Point_Y = 32;
Line22.Second_Point_X = 50;
Line22.Second_Point_Y = 32;
Line22.Pen_Width = 3;
Line22.Pen_Color = 0x1515;

Line23.OwnerScreen = &ProfilingScreen;
Line23.Order = 41;
Line23.Visible = 0;
Line23.Opacity = 255;
Line23.Tag = 255;
Line23.First_Point_X = 40;
Line23.First_Point_Y = 32;
Line23.Second_Point_X = 50;
Line23.Second_Point_Y = 32;
Line23.Pen_Width = 3;
Line23.Pen_Color = 0x1515;

Line24.OwnerScreen = &ProfilingScreen;
Line24.Order = 42;
Line24.Visible = 0;
Line24.Opacity = 255;
Line24.Tag = 255;
Line24.First_Point_X = 40;
Line24.First_Point_Y = 32;
Line24.Second_Point_X = 50;
Line24.Second_Point_Y = 32;
Line24.Pen_Width = 3;
Line24.Pen_Color = 0x1515;

Line25.OwnerScreen = &ProfilingScreen;
Line25.Order = 43;
Line25.Visible = 0;
Line25.Opacity = 255;
Line25.Tag = 255;
Line25.First_Point_X = 40;
Line25.First_Point_Y = 32;
Line25.Second_Point_X = 50;
Line25.Second_Point_Y = 32;
Line25.Pen_Width = 3;
Line25.Pen_Color = 0x1515;

Line26.OwnerScreen = &ProfilingScreen;
Line26.Order = 44;
Line26.Visible = 0;
Line26.Opacity = 255;
Line26.Tag = 255;
Line26.First_Point_X = 40;
Line26.First_Point_Y = 32;
Line26.Second_Point_X = 50;
Line26.Second_Point_Y = 32;
Line26.Pen_Width = 3;
Line26.Pen_Color = 0x1515;

Line27.OwnerScreen = &ProfilingScreen;
Line27.Order = 45;
Line27.Visible = 0;
Line27.Opacity = 255;
Line27.Tag = 255;
Line27.First_Point_X = 40;
Line27.First_Point_Y = 32;
Line27.Second_Point_X = 50;
Line27.Second_Point_Y = 32;
Line27.Pen_Width = 3;
Line27.Pen_Color = 0x1515;

Line28.OwnerScreen = &ProfilingScreen;
Line28.Order = 46;
Line28.Visible = 0;
Line28.Opacity = 255;
Line28.Tag = 255;
Line28.First_Point_X = 40;
Line28.First_Point_Y = 32;
Line28.Second_Point_X = 50;
Line28.Second_Point_Y = 32;
Line28.Pen_Width = 3;
Line28.Pen_Color = 0x1515;

Line29.OwnerScreen = &ProfilingScreen;
Line29.Order = 47;
Line29.Visible = 0;
Line29.Opacity = 255;
Line29.Tag = 255;
Line29.First_Point_X = 40;
Line29.First_Point_Y = 32;
Line29.Second_Point_X = 50;
Line29.Second_Point_Y = 32;
Line29.Pen_Width = 3;
Line29.Pen_Color = 0x1515;

Line30.OwnerScreen = &ProfilingScreen;
Line30.Order = 48;
Line30.Visible = 0;
Line30.Opacity = 255;
Line30.Tag = 255;
Line30.First_Point_X = 40;
Line30.First_Point_Y = 32;
Line30.Second_Point_X = 50;
Line30.Second_Point_Y = 32;
Line30.Pen_Width = 3;
Line30.Pen_Color = 0x1515;

Line31.OwnerScreen = &ProfilingScreen;
Line31.Order = 49;
Line31.Visible = 0;
```

```
Line31.Opacity = 255;
Line31.Tag = 255;
Line31.First_Point_X = 40;
Line31.First_Point_Y = 32;
Line31.Second_Point_X = 50;
Line31.Second_Point_Y = 32;
Line31.Pen_Width = 3;
Line31.Pen_Color = 0x1515;

Line32.OwnerScreen = &ProfilingScreen;
Line32.Order = 50;
Line32.Visible = 0;
Line32.Opacity = 255;
Line32.Tag = 255;
Line32.First_Point_X = 40;
Line32.First_Point_Y = 32;
Line32.Second_Point_X = 50;
Line32.Second_Point_Y = 32;
Line32.Pen_Width = 3;
Line32.Pen_Color = 0x1515;

Line33.OwnerScreen = &ProfilingScreen;
Line33.Order = 51;
Line33.Visible = 0;
Line33.Opacity = 255;
Line33.Tag = 255;
Line33.First_Point_X = 40;
Line33.First_Point_Y = 32;
Line33.Second_Point_X = 50;
Line33.Second_Point_Y = 32;
Line33.Pen_Width = 3;
Line33.Pen_Color = 0x1515;

Line34.OwnerScreen = &ProfilingScreen;
Line34.Order = 52;
Line34.Visible = 0;
Line34.Opacity = 255;
Line34.Tag = 255;
Line34.First_Point_X = 40;
Line34.First_Point_Y = 32;
Line34.Second_Point_X = 50;
Line34.Second_Point_Y = 32;
Line34.Pen_Width = 3;
Line34.Pen_Color = 0x1515;

Line35.OwnerScreen = &ProfilingScreen;
Line35.Order = 53;
Line35.Visible = 0;
Line35.Opacity = 255;
Line35.Tag = 255;
Line35.First_Point_X = 40;
Line35.First_Point_Y = 32;
Line35.Second_Point_X = 50;
Line35.Second_Point_Y = 32;
Line35.Pen_Width = 3;
Line35.Pen_Color = 0x1515;

Line36.OwnerScreen = &ProfilingScreen;
Line36.Order = 54;
Line36.Visible = 0;
Line36.Opacity = 255;
Line36.Tag = 255;
Line36.First_Point_X = 40;
Line36.First_Point_Y = 32;
Line36.Second_Point_X = 50;
Line36.Second_Point_Y = 32;
Line36.Pen_Width = 3;
Line36.Pen_Color = 0x1515;

Line37.OwnerScreen = &ProfilingScreen;
Line37.Order = 55;
Line37.Visible = 0;
Line37.Opacity = 255;
Line37.Tag = 255;
Line37.First_Point_X = 40;
Line37.First_Point_Y = 32;
Line37.Second_Point_X = 50;
Line37.Second_Point_Y = 32;
Line37.Pen_Width = 3;
Line37.Pen_Color = 0x1515;

Line38.OwnerScreen = &ProfilingScreen;
Line38.Order = 56;
Line38.Visible = 0;
Line38.Opacity = 255;
Line38.Tag = 255;
Line38.First_Point_X = 40;
Line38.First_Point_Y = 32;
Line38.Second_Point_X = 50;
Line38.Second_Point_Y = 32;
Line38.Pen_Width = 3;
Line38.Pen_Color = 0x1515;

Line39.OwnerScreen = &ProfilingScreen;
Line39.Order = 57;
Line39.Visible = 0;
Line39.Opacity = 255;
Line39.Tag = 255;
Line39.First_Point_X = 40;
Line39.First_Point_Y = 32;
Line39.Second_Point_X = 50;
Line39.Second_Point_Y = 32;
Line39.Pen_Width = 3;
Line39.Pen_Color = 0x1515;

Line40.OwnerScreen = &ProfilingScreen;
Line40.Order = 58;
Line40.Visible = 0;
Line40.Opacity = 255;
Line40.Tag = 255;
Line40.First_Point_X = 40;
Line40.First_Point_Y = 32;
Line40.Second_Point_X = 50;
Line40.Second_Point_Y = 32;
Line40.Pen_Width = 3;
Line40.Pen_Color = 0x1515;

Line41.OwnerScreen = &ProfilingScreen;
Line41.Order = 59;
Line41.Visible = 0;
Line41.Opacity = 255;
Line41.Tag = 255;
Line41.First_Point_X = 40;
Line41.First_Point_Y = 32;
Line41.Second_Point_X = 50;
Line41.Second_Point_Y = 32;
Line41.Pen_Width = 3;
Line41.Pen_Color = 0x1515;

Line42.OwnerScreen = &ProfilingScreen;
Line42.Order = 60;
Line42.Visible = 0;
Line42.Opacity = 255;
Line42.Tag = 255;
Line42.First_Point_X = 40;
Line42.First_Point_Y = 32;
Line42.Second_Point_X = 50;
Line42.Second_Point_Y = 32;
Line42.Pen_Width = 3;
Line42.Pen_Color = 0x1515;
```

```
Line43.OwnerScreen = &ProfilingScreen;
Line43.Order = 61;
Line43.Visible = 0;
Line43.Opacity = 255;
Line43.Tag = 255;
Line43.First_Point_X = 40;
Line43.First_Point_Y = 32;
Line43.Second_Point_X = 50;
Line43.Second_Point_Y = 32;
Line43.Pen_Width = 3;
Line43.Pen_Color = 0x1515;

Line44.OwnerScreen = &ProfilingScreen;
Line44.Order = 62;
Line44.Visible = 0;
Line44.Opacity = 255;
Line44.Tag = 255;
Line44.First_Point_X = 40;
Line44.First_Point_Y = 32;
Line44.Second_Point_X = 50;
Line44.Second_Point_Y = 32;
Line44.Pen_Width = 3;
Line44.Pen_Color = 0x1515;

Line45.OwnerScreen = &ProfilingScreen;
Line45.Order = 63;
Line45.Visible = 0;
Line45.Opacity = 255;
Line45.Tag = 255;
Line45.First_Point_X = 40;
Line45.First_Point_Y = 32;
Line45.Second_Point_X = 50;
Line45.Second_Point_Y = 32;
Line45.Pen_Width = 3;
Line45.Pen_Color = 0x1515;

Line46.OwnerScreen = &ProfilingScreen;
Line46.Order = 64;
Line46.Visible = 0;
Line46.Opacity = 255;
Line46.Tag = 255;
Line46.First_Point_X = 40;
Line46.First_Point_Y = 32;
Line46.Second_Point_X = 50;
Line46.Second_Point_Y = 32;
Line46.Pen_Width = 3;
Line46.Pen_Color = 0x1515;

Line47.OwnerScreen = &ProfilingScreen;
Line47.Order = 65;
Line47.Visible = 0;
Line47.Opacity = 255;
Line47.Tag = 255;
Line47.First_Point_X = 40;
Line47.First_Point_Y = 32;
Line47.Second_Point_X = 50;
Line47.Second_Point_Y = 32;
Line47.Pen_Width = 3;
Line47.Pen_Color = 0x1515;

Line48.OwnerScreen = &ProfilingScreen;
Line48.Order = 66;
Line48.Visible = 0;
Line48.Opacity = 255;
Line48.Tag = 255;
Line48.First_Point_X = 40;
Line48.First_Point_Y = 32;
Line48.Second_Point_X = 50;
Line48.Second_Point_Y = 32;
Line48.Pen_Width = 3;
Line48.Pen_Color = 0x1515;

Line49.OwnerScreen = &ProfilingScreen;
Line49.Order = 67;
Line49.Visible = 0;
Line49.Opacity = 255;
Line49.Tag = 255;
Line49.First_Point_X = 40;
Line49.First_Point_Y = 32;
Line49.Second_Point_X = 50;
Line49.Second_Point_Y = 32;
Line49.Pen_Width = 3;
Line49.Pen_Color = 0x1515;

Line50.OwnerScreen = &ProfilingScreen;
Line50.Order = 68;
Line50.Visible = 0;
Line50.Opacity = 255;
Line50.Tag = 255;
Line50.First_Point_X = 40;
Line50.First_Point_Y = 32;
Line50.Second_Point_X = 50;
Line50.Second_Point_Y = 32;
Line50.Pen_Width = 3;
Line50.Pen_Color = 0x1515;

EveText1.OwnerScreen = &ProfilingScreen;
EveText1.Order = 69;
EveText1.Visible = 1;
EveText1.Opacity = 255;
EveText1.Tag = 255;
EveText1.Left = 275;
EveText1.Top = 160;
EveText1.Width = 83;
EveText1.Height = 15;
EveText1.Caption = EveText1_Caption;
EveText1.TextAlign = taNone;
EveText1.FontName = 26;
EveText1.Font_Color = 0x0148;
EveText1.FontHandle = 26;
EveText1.Source = -1UL;
EveText1.Active = 1;
EveText1.OnUp = 0;
EveText1.OnDown = 0;
EveText1.OnClick = 0;
EveText1.OnPress = 0;

EveButtonProfCreateFile.OwnerScreen = &ProfilingScreen;
EveButtonProfCreateFile.Order = 70;
EveButtonProfCreateFile.Visible = 1;
EveButtonProfCreateFile.Opacity = 255;
EveButtonProfCreateFile.Tag = 255;
EveButtonProfCreateFile.Left = 274;
EveButtonProfCreateFile.Top = 214;
EveButtonProfCreateFile.Width = 200;
EveButtonProfCreateFile.Height = 48;
EveButtonProfCreateFile.Color = 0x03DA;
EveButtonProfCreateFile.Press_Color = 0x7E3F;
EveButtonProfCreateFile.ColorTo = 0x7E3F;
EveButtonProfCreateFile.Press_ColorTo = 0x03DA;
EveButtonProfCreateFile.Caption =
EveButtonProfCreateFile_Caption;
EveButtonProfCreateFile.FontName = 27;
EveButtonProfCreateFile.Font_Color = 0xFFFF;
EveButtonProfCreateFile.FontHandle = 27;
EveButtonProfCreateFile.Source = -1UL;
EveButtonProfCreateFile.Flat = 0;
EveButtonProfCreateFile.Active = 1;
EveButtonProfCreateFile.OnUp = 0;
```

```
 EveButtonProfCreateFile.OnDown = 0;
 EveButtonProfCreateFile.OnClick =
&EveButtonProfCreateFile_OnClick;
 EveButtonProfCreateFile.OnPress = 0;

 EveButtonProfCreateFile_OnClick.Action =
EveButtonProfCreateFileOnClick;
 EveButtonProfCreateFile_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonProfCreateFile_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonProfCreateFile_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonProfCreateFile_OnClick.Sound.Volume = 255;

 EveTextProfFileName.OwnerScreen = &ProfilingScreen;
 EveTextProfFileName.Order = 72;
 EveTextProfFileName.Visible = 1;
 EveTextProfFileName.Opacity = 255;
 EveTextProfFileName.Tag = 255;
 EveTextProfFileName.Left = 305;
 EveTextProfFileName.Top = 178;
 EveTextProfFileName.Width = 27;
 EveTextProfFileName.Height = 15;
 EveTextProfFileName.Caption = EveTextProfFileName_Caption;
 EveTextProfFileName.TextAlign = taNone;
 EveTextProfFileName.FontName = 26;
 EveTextProfFileName.Font_Color = 0x0148;
 EveTextProfFileName.FontHandle = 26;
 EveTextProfFileName.Source = -1UL;
 EveTextProfFileName.Active = 0;
 EveTextProfFileName.OnUp = 0;
 EveTextProfFileName.OnDown = 0;
 EveTextProfFileName.OnClick = 0;
 EveTextProfFileName.OnPress = 0;

 BoxProfWaitingForCam.OwnerScreen = &ProfilingScreen;
 BoxProfWaitingForCam.Order = 74;
 BoxProfWaitingForCam.Visible = 0;
 BoxProfWaitingForCam.Opacity = 196;
 BoxProfWaitingForCam.Tag = 255;
 BoxProfWaitingForCam.Left = 2;
 BoxProfWaitingForCam.Top = 4;
 BoxProfWaitingForCam.Width = 476;
 BoxProfWaitingForCam.Height = 266;
 BoxProfWaitingForCam.Pen_Width = 1;
 BoxProfWaitingForCam.Pen_Color = 0x0000;
 BoxProfWaitingForCam.Color = 0xFFFF;
 BoxProfWaitingForCam.Press_Color = 0xFFFF;
 BoxProfWaitingForCam.ColorTo = 0xFFFF;
 BoxProfWaitingForCam.Press_ColorTo = 0xC618;
 BoxProfWaitingForCam.Gradient =
_FT800_BRUSH_GR_NONE;
 BoxProfWaitingForCam.Active = 0;
 BoxProfWaitingForCam.OnUp = 0;
 BoxProfWaitingForCam.OnDown = 0;
 BoxProfWaitingForCam.OnClick = 0;
 BoxProfWaitingForCam.OnPress = 0;

 EveTextProfWaitingForCam.OwnerScreen = &ProfilingScreen;
 EveTextProfWaitingForCam.Order = 75;
 EveTextProfWaitingForCam.Visible = 0;
 EveTextProfWaitingForCam.Opacity = 255;
 EveTextProfWaitingForCam.Tag = 255;
 EveTextProfWaitingForCam.Left = 106;
 EveTextProfWaitingForCam.Top = 75;
 EveTextProfWaitingForCam.Width = 265;
 EveTextProfWaitingForCam.Height = 25;
 EveTextProfWaitingForCam.Caption =
EveTextProfWaitingForCam_Caption;
EveTextProfWaitingForCam.TextAlign = taNone;
EveTextProfWaitingForCam.FontName = 29;
EveTextProfWaitingForCam.Font_Color = 0x047F;
EveTextProfWaitingForCam.FontHandle = 29;
EveTextProfWaitingForCam.Source = -1UL;
EveTextProfWaitingForCam.Active = 0;
EveTextProfWaitingForCam.OnUp = 0;
EveTextProfWaitingForCam.OnDown = 0;
EveTextProfWaitingForCam.OnClick = 0;
EveTextProfWaitingForCam.OnPress = 0;

EveButtonFNfilename.OwnerScreen = &SettingScreen;
EveButtonFNfilename.Order = 0;
EveButtonFNfilename.Visible = 1;
EveButtonFNfilename.Opacity = 255;
EveButtonFNfilename.Tag = 255;
EveButtonFNfilename.Left = 310;
EveButtonFNfilename.Top = 4;
EveButtonFNfilename.Width = 108;
EveButtonFNfilename.Height = 26;
EveButtonFNfilename.Color = 0xD6BA;
EveButtonFNfilename.Press_Color = 0x7E3F;
EveButtonFNfilename.ColorTo = 0xC618;
EveButtonFNfilename.Press_ColorTo = 0x03DA;
EveButtonFNfilename.Caption = EveButtonFNfilename_Caption;
EveButtonFNfilename.FontName = 28;
EveButtonFNfilename.Font_Color = 0x0000;
EveButtonFNfilename.FontHandle = 28;
EveButtonFNfilename.Source = -1UL;
EveButtonFNfilename.Flat = 1;
EveButtonFNfilename.Active = 0;
EveButtonFNfilename.OnUp = 0;
EveButtonFNfilename.OnDown = 0;
EveButtonFNfilename.OnClick = 0;
EveButtonFNfilename.OnPress = 0;

EveKeys1.OwnerScreen = &SettingScreen;
EveKeys1.Order = 2;
EveKeys1.Visible = 1;
EveKeys1.Opacity = 255;
EveKeys1.Tag = 255;
EveKeys1.Left = 9;
EveKeys1.Top = 93;
EveKeys1.Width = 463;
EveKeys1.Height = 40;
EveKeys1.Color = 0x03DA;
EveKeys1.Press_Color = 0x7E3F;
EveKeys1.ColorTo = 0x7E3F;
EveKeys1.Press_ColorTo = 0x03DA;
EveKeys1.Caption = EveKeys1_Caption;
EveKeys1.FontName = 27;
EveKeys1.Font_Color = 0xFFFF;
EveKeys1.FontHandle = 27;
EveKeys1.Source = -1UL;
EveKeys1.Flat = 0;
EveKeys1.AutoSize = 0;
EveKeys1.Active = 1;
EveKeys1.OnUp = 0;
EveKeys1.OnDown = 0;
EveKeys1.OnClick = 0;
EveKeys1.OnPress = 0;

EveKeys2.OwnerScreen = &SettingScreen;
EveKeys2.Order = 3;
EveKeys2.Visible = 1;
EveKeys2.Opacity = 255;
EveKeys2.Tag = 255;
EveKeys2.Left = 9;
EveKeys2.Top = 136;
EveKeys2.Width = 464;
```

```
EveKeys2.Height = 40;
EveKeys2.Color = 0x03DA;
EveKeys2.Press_Color = 0x7E3F;
EveKeys2.ColorTo = 0x7E3F;
EveKeys2.Press_ColorTo = 0x03DA;
EveKeys2.Caption = EveKeys2_Caption;
EveKeys2.FontName = 27;
EveKeys2.Font_Color = 0xFFFF;
EveKeys2.FontHandle = 27;
EveKeys2.Source = -1UL;
EveKeys2.Flat = 0;
EveKeys2.AutoSize = 0;
EveKeys2.Active = 1;
EveKeys2.OnUp = 0;
EveKeys2.OnDown = 0;
EveKeys2.OnClick = 0;
EveKeys2.OnPress = 0;

EveKeys3.OwnerScreen = &SettingScreen;
EveKeys3.Order = 4;
EveKeys3.Visible = 1;
EveKeys3.Opacity = 255;
EveKeys3.Tag = 255;
EveKeys3.Left = 9;
EveKeys3.Top = 179;
EveKeys3.Width = 464;
EveKeys3.Height = 40;
EveKeys3.Color = 0x03DA;
EveKeys3.Press_Color = 0x7E3F;
EveKeys3.ColorTo = 0x7E3F;
EveKeys3.Press_ColorTo = 0x03DA;
EveKeys3.Caption = EveKeys3_Caption;
EveKeys3.FontName = 27;
EveKeys3.Font_Color = 0xFFFF;
EveKeys3.FontHandle = 27;
EveKeys3.Source = -1UL;
EveKeys3.Flat = 0;
EveKeys3.AutoSize = 0;
EveKeys3.Active = 1;
EveKeys3.OnUp = 0;
EveKeys3.OnDown = 0;
EveKeys3.OnClick = 0;
EveKeys3.OnPress = 0;

EveKeys4.OwnerScreen = &SettingScreen;
EveKeys4.Order = 5;
EveKeys4.Visible = 1;
EveKeys4.Opacity = 255;
EveKeys4.Tag = 255;
EveKeys4.Left = 62;
EveKeys4.Top = 222;
EveKeys4.Width = 410;
EveKeys4.Height = 40;
EveKeys4.Color = 0x03DA;
EveKeys4.Press_Color = 0x7E3F;
EveKeys4.ColorTo = 0x7E3F;
EveKeys4.Press_ColorTo = 0x03DA;
EveKeys4.Caption = EveKeys4_Caption;
EveKeys4.FontName = 27;
EveKeys4.Font_Color = 0xFFFF;
EveKeys4.FontHandle = 27;
EveKeys4.Source = -1UL;
EveKeys4.Flat = 0;
EveKeys4.AutoSize = 0;
EveKeys4.Active = 1;
EveKeys4.OnUp = 0;
EveKeys4.OnDown = 0;
EveKeys4.OnClick = 0;
EveKeys4.OnPress = 0;

EveButtonFNBackspace.OwnerScreen = &SettingScreen;
EveButtonFNBackspace.Order = 8;
EveButtonFNBackspace.Visible = 1;
EveButtonFNBackspace.Opacity = 255;
EveButtonFNBackspace.Tag = 255;
EveButtonFNBackspace.Left = 10;
EveButtonFNBackspace.Top = 38;
EveButtonFNBackspace.Width = 102;
EveButtonFNBackspace.Height = 39;
EveButtonFNBackspace.Color = 0x4AEC;
EveButtonFNBackspace.Press_Color = 0x7E3F;
EveButtonFNBackspace.ColorTo = 0x7E3F;
EveButtonFNBackspace.Press_ColorTo = 0x03DA;
EveButtonFNBackspace.Caption =
EveButtonFNBackspace_Caption;
EveButtonFNBackspace.FontName = 27;
EveButtonFNBackspace.Font_Color = 0xFFFF;
EveButtonFNBackspace.FontHandle = 27;
EveButtonFNBackspace.Source = -1UL;
EveButtonFNBackspace.Flat = 0;
EveButtonFNBackspace.Active = 1;
EveButtonFNBackspace.OnUp = 0;
EveButtonFNBackspace.OnDown = 0;
EveButtonFNBackspace.OnClick =
&EveButtonFNBackspace_OnClick;
EveButtonFNBackspace.OnPress = 0;

EveButtonFNBackspace_OnClick.Action =
EveButtonFNBackspaceOnClick;
EveButtonFNBackspace_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
EveButtonFNBackspace_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
EveButtonFNBackspace_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
EveButtonFNBackspace_OnClick.Sound.Volume = 255;

EveButtonFNClear.OwnerScreen = &SettingScreen;
EveButtonFNClear.Order = 9;
EveButtonFNClear.Visible = 1;
EveButtonFNClear.Opacity = 255;
EveButtonFNClear.Tag = 255;
EveButtonFNClear.Left = 118;
EveButtonFNClear.Top = 38;
EveButtonFNClear.Width = 104;
EveButtonFNClear.Height = 40;
EveButtonFNClear.Color = 0x03DA;
EveButtonFNClear.Press_Color = 0x7E3F;
EveButtonFNClear.ColorTo = 0x7E3F;
EveButtonFNClear.Press_ColorTo = 0x03DA;
EveButtonFNClear.Caption = EveButtonFNClear_Caption;
EveButtonFNClear.FontName = 27;
EveButtonFNClear.Font_Color = 0xFFFF;
EveButtonFNClear.FontHandle = 27;
EveButtonFNClear.Source = -1UL;
EveButtonFNClear.Flat = 0;
EveButtonFNClear.Active = 1;
EveButtonFNClear.OnUp = 0;
EveButtonFNClear.OnDown = 0;
EveButtonFNClear.OnClick = &EveButtonFNClear_OnClick;
EveButtonFNClear.OnPress = 0;

EveButtonFNClear_OnClick.Action =
EveButtonFNClearOnClick;
EveButtonFNClear_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
EveButtonFNClear_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
EveButtonFNClear_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
```

```
 EveButtonFNClear_OnClick.Sound.Volume = 255;

 EveButtonFNConfirm.OwnerScreen = &SettingScreen;
 EveButtonFNConfirm.Order = 10;
 EveButtonFNConfirm.Visible = 1;
 EveButtonFNConfirm.Opacity = 255;
 EveButtonFNConfirm.Tag = 255;
 EveButtonFNConfirm.Left = 358;
 EveButtonFNConfirm.Top = 38;
 EveButtonFNConfirm.Width = 108;
 EveButtonFNConfirm.Height = 38;
 EveButtonFNConfirm.Color = 0x0583;
 EveButtonFNConfirm.Press_Color = 0x7E3F;
 EveButtonFNConfirm.ColorTo = 0x7E3F;
 EveButtonFNConfirm.Press_ColorTo = 0x03DA;
 EveButtonFNConfirm.Caption = EveButtonFNConfirm_Caption;
 EveButtonFNConfirm.FontName = 27;
 EveButtonFNConfirm.Font_Color = 0xFFFF;
 EveButtonFNConfirm.FontHandle = 27;
 EveButtonFNConfirm.Source = -1UL;
 EveButtonFNConfirm.Flat = 0;
 EveButtonFNConfirm.Active = 1;
 EveButtonFNConfirm.OnUp = 0;
 EveButtonFNConfirm.OnDown = 0;
 EveButtonFNConfirm.OnClick =
&EveButtonFNConfirm_OnClick;
 EveButtonFNConfirm.OnPress = 0;

 EveButtonFNConfirm_OnClick.Action =
EveButtonFNConfirmOnClick;
 EveButtonFNConfirm_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonFNConfirm_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonFNConfirm_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonFNConfirm_OnClick.Sound.Volume = 255;

 EveButtonFNGoBack.OwnerScreen = &SettingScreen;
 EveButtonFNGoBack.Order = 11;
 EveButtonFNGoBack.Visible = 1;
 EveButtonFNGoBack.Opacity = 255;
 EveButtonFNGoBack.Tag = 255;
 EveButtonFNGoBack.Left = 232;
 EveButtonFNGoBack.Top = 38;
 EveButtonFNGoBack.Width = 118;
 EveButtonFNGoBack.Height = 40;
 EveButtonFNGoBack.Color = 0xB003;
 EveButtonFNGoBack.Press_Color = 0x7E3F;
 EveButtonFNGoBack.ColorTo = 0xF432;
 EveButtonFNGoBack.Press_ColorTo = 0x03DA;
 EveButtonFNGoBack.Caption = EveButtonFNGoBack_Caption;
 EveButtonFNGoBack.FontName = 27;
 EveButtonFNGoBack.Font_Color = 0xFFFF;
 EveButtonFNGoBack.FontHandle = 27;
 EveButtonFNGoBack.Source = -1UL;
 EveButtonFNGoBack.Flat = 0;
 EveButtonFNGoBack.Active = 1;
 EveButtonFNGoBack.OnUp = 0;
 EveButtonFNGoBack.OnDown = 0;
 EveButtonFNGoBack.OnClick =
&EveButtonFNGoBack_OnClick;
 EveButtonFNGoBack.OnPress = 0;

 EveButtonFNGoBack_OnClick.Action =
EveButtonFNGoBackOnClick;
 EveButtonFNGoBack_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonFNGoBack_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonFNGoBack_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonFNGoBack_OnClick.Sound.Volume = 255;

 EveButtonFNShift.OwnerScreen = &SettingScreen;
 EveButtonFNShift.Order = 12;
 EveButtonFNShift.Visible = 1;
 EveButtonFNShift.Opacity = 255;
 EveButtonFNShift.Tag = 255;
 EveButtonFNShift.Left = 10;
 EveButtonFNShift.Top = 222;
 EveButtonFNShift.Width = 48;
 EveButtonFNShift.Height = 39;
 EveButtonFNShift.Color = 0x042B;
 EveButtonFNShift.Press_Color = 0x7E3F;
 EveButtonFNShift.ColorTo = 0x87BA;
 EveButtonFNShift.Press_ColorTo = 0x03DA;
 EveButtonFNShift.Caption = EveButtonFNShift_Caption;
 EveButtonFNShift.FontName = 27;
 EveButtonFNShift.Font_Color = 0xFFFF;
 EveButtonFNShift.FontHandle = 27;
 EveButtonFNShift.Source = -1UL;
 EveButtonFNShift.Flat = 0;
 EveButtonFNShift.Active = 1;
 EveButtonFNShift.OnUp = 0;
 EveButtonFNShift.OnDown = 0;
 EveButtonFNShift.OnClick = &EveButtonFNShift_OnClick;
 EveButtonFNShift.OnPress = 0;

 EveButtonFNShift_OnClick.Action = EveButtonFNShiftOnClick;
 EveButtonFNShift_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonFNShift_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
 EveButtonFNShift_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
 EveButtonFNShift_OnClick.Sound.Volume = 255;

 EveButtonSUNewRun.OwnerScreen = &SummaryScreen;
 EveButtonSUNewRun.Order = 7;
 EveButtonSUNewRun.Visible = 1;
 EveButtonSUNewRun.Opacity = 255;
 EveButtonSUNewRun.Tag = 255;
 EveButtonSUNewRun.Left = 337;
 EveButtonSUNewRun.Top = 198;
 EveButtonSUNewRun.Width = 122;
 EveButtonSUNewRun.Height = 52;
 EveButtonSUNewRun.Color = 0x03DA;
 EveButtonSUNewRun.Press_Color = 0x7E3F;
 EveButtonSUNewRun.ColorTo = 0x7E3F;
 EveButtonSUNewRun.Press_ColorTo = 0x03DA;
 EveButtonSUNewRun.Caption = EveButtonSUNewRun_Caption;
 EveButtonSUNewRun.FontName = 27;
 EveButtonSUNewRun.Font_Color = 0xFFFF;
 EveButtonSUNewRun.FontHandle = 27;
 EveButtonSUNewRun.Source = -1UL;
 EveButtonSUNewRun.Flat = 0;
 EveButtonSUNewRun.Active = 1;
 EveButtonSUNewRun.OnUp = 0;
 EveButtonSUNewRun.OnDown = 0;
 EveButtonSUNewRun.OnClick =
&EveButtonSUNewRun_OnClick;
 EveButtonSUNewRun.OnPress = 0;

 EveButtonSUNewRun_OnClick.Action =
EveButtonSUNewRunOnClick;
 EveButtonSUNewRun_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
 EveButtonSUNewRun_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
```

```
EveButtonSUNewRun_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
EveButtonSUNewRun_OnClick.Sound.Volume = 255;

EveButtonSUHelp.OwnerScreen = &SummaryScreen;
EveButtonSUHelp.Order = 8;
EveButtonSUHelp.Visible = 1;
EveButtonSUHelp.Opacity = 255;
EveButtonSUHelp.Tag = 255;
EveButtonSUHelp.Left = 337;
EveButtonSUHelp.Top = 142;
EveButtonSUHelp.Width = 122;
EveButtonSUHelp.Height = 52;
EveButtonSUHelp.Color = 0x03DA;
EveButtonSUHelp.Press_Color = 0x7E3F;
EveButtonSUHelp.ColorTo = 0x7E3F;
EveButtonSUHelp.Press_ColorTo = 0x03DA;
EveButtonSUHelp.Caption = EveButtonSUHelp_Caption;
EveButtonSUHelp.FontName = 27;
EveButtonSUHelp.Font_Color = 0xFFFF;
EveButtonSUHelp.FontHandle = 27;
EveButtonSUHelp.Source = -1UL;
EveButtonSUHelp.Flat = 0;
EveButtonSUHelp.Active = 1;
EveButtonSUHelp.OnUp = 0;
EveButtonSUHelp.OnDown = 0;
EveButtonSUHelp.OnClick = &EveButtonSUHelp_OnClick;
EveButtonSUHelp.OnPress = 0;

EveButtonSUHelp_OnClick.Action = EveButtonSUHelpOnClick;
EveButtonSUHelp_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
EveButtonSUHelp_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
EveButtonSUHelp_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
EveButtonSUHelp_OnClick.Sound.Volume = 255;

EveButtonSUAbout.OwnerScreen = &SummaryScreen;
EveButtonSUAbout.Order = 9;
EveButtonSUAbout.Visible = 1;
EveButtonSUAbout.Opacity = 255;
EveButtonSUAbout.Tag = 255;
EveButtonSUAbout.Left = 337;
EveButtonSUAbout.Top = 80;
EveButtonSUAbout.Width = 122;
EveButtonSUAbout.Height = 57;
EveButtonSUAbout.Color = 0x03DA;
EveButtonSUAbout.Press_Color = 0x7E3F;
EveButtonSUAbout.ColorTo = 0x7E3F;
EveButtonSUAbout.Press_ColorTo = 0x03DA;
EveButtonSUAbout.Caption = EveButtonSUAbout_Caption;
EveButtonSUAbout.FontName = 27;
EveButtonSUAbout.Font_Color = 0xFFFF;
EveButtonSUAbout.FontHandle = 27;
EveButtonSUAbout.Source = -1UL;
EveButtonSUAbout.Flat = 0;
EveButtonSUAbout.Active = 1;
EveButtonSUAbout.OnUp = 0;
EveButtonSUAbout.OnDown = 0;
EveButtonSUAbout.OnClick = &EveButtonSUAbout_OnClick;
EveButtonSUAbout.OnPress = 0;

EveButtonSUAbout_OnClick.Action =
EveButtonSUAboutOnClick;
EveButtonSUAbout_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
EveButtonSUAbout_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
EveButtonSUAbout_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
EveButtonSUAbout_OnClick.Sound.Volume = 255;

EveTextSUtime.OwnerScreen = &SummaryScreen;
EveTextSUtime.Order = 10;
EveTextSUtime.Visible = 1;
EveTextSUtime.Opacity = 255;
EveTextSUtime.Tag = 255;
EveTextSUtime.Left = 178;
EveTextSUtime.Top = 39;
EveTextSUtime.Width = 8;
EveTextSUtime.Height = 17;
EveTextSUtime.Caption = EveTextSUtime_Caption;
EveTextSUtime.TextAlign = taNone;
EveTextSUtime.FontName = 27;
EveTextSUtime.Font_Color = 0x0148;
EveTextSUtime.FontHandle = 27;
EveTextSUtime.Source = -1UL;
EveTextSUtime.Active = 1;
EveTextSUtime.OnUp = 0;
EveTextSUtime.OnDown = 0;
EveTextSUtime.OnClick = 0;
EveTextSUtime.OnPress = 0;

EveTextSUdistance.OwnerScreen = &SummaryScreen;
EveTextSUdistance.Order = 11;
EveTextSUdistance.Visible = 1;
EveTextSUdistance.Opacity = 255;
EveTextSUdistance.Tag = 255;
EveTextSUdistance.Left = 214;
EveTextSUdistance.Top = 80;
EveTextSUdistance.Width = 8;
EveTextSUdistance.Height = 17;
EveTextSUdistance.Caption = EveTextSUdistance_Caption;
EveTextSUdistance.TextAlign = taNone;
EveTextSUdistance.FontName = 27;
EveTextSUdistance.Font_Color = 0x0148;
EveTextSUdistance.FontHandle = 27;
EveTextSUdistance.Source = -1UL;
EveTextSUdistance.Active = 1;
EveTextSUdistance.OnUp = 0;
EveTextSUdistance.OnDown = 0;
EveTextSUdistance.OnClick = 0;
EveTextSUdistance.OnPress = 0;

EveTextSUSpeed.OwnerScreen = &SummaryScreen;
EveTextSUSpeed.Order = 12;
EveTextSUSpeed.Visible = 1;
EveTextSUSpeed.Opacity = 255;
EveTextSUSpeed.Tag = 255;
EveTextSUSpeed.Left = 271;
EveTextSUSpeed.Top = 166;
EveTextSUSpeed.Width = 8;
EveTextSUSpeed.Height = 17;
EveTextSUSpeed.Caption = EveTextSUSpeed_Caption;
EveTextSUSpeed.TextAlign = taNone;
EveTextSUSpeed.FontName = 27;
EveTextSUSpeed.Font_Color = 0x0148;
EveTextSUSpeed.FontHandle = 27;
EveTextSUSpeed.Source = -1UL;
EveTextSUSpeed.Active = 1;
EveTextSUSpeed.OnUp = 0;
EveTextSUSpeed.OnDown = 0;
EveTextSUSpeed.OnClick = 0;
EveTextSUSpeed.OnPress = 0;

EveTextSUFilename.OwnerScreen = &SummaryScreen;
EveTextSUFilename.Order = 13;
EveTextSUFilename.Visible = 1;
```

```
EveTextSUFilename.Opacity = 255;
EveTextSUFilename.Tag = 255;
EveTextSUFilename.Left = 162;
EveTextSUFilename.Top = 125;
EveTextSUFilename.Width = 32;
EveTextSUFilename.Height = 17;
EveTextSUFilename.Caption = EveTextSUFilename_Caption;
EveTextSUFilename.TextAlign = taNone;
EveTextSUFilename.FontName = 27;
EveTextSUFilename.Font_Color = 0x0148;
EveTextSUFilename.FontHandle = 27;
EveTextSUFilename.Source = -1UL;
EveTextSUFilename.Active = 1;
EveTextSUFilename.OnUp = 0;
EveTextSUFilename.OnDown = 0;
EveTextSUFilename.OnClick = 0;
EveTextSUFilename.OnPress = 0;

EveTextSUAverageSpeed.OwnerScreen = &SummaryScreen;
EveTextSUAverageSpeed.Order = 14;
EveTextSUAverageSpeed.Visible = 1;
EveTextSUAverageSpeed.Opacity = 255;
EveTextSUAverageSpeed.Tag = 255;
EveTextSUAverageSpeed.Left = 184;
EveTextSUAverageSpeed.Top = 206;
EveTextSUAverageSpeed.Width = 8;
EveTextSUAverageSpeed.Height = 17;
EveTextSUAverageSpeed.Caption =
EveTextSUAverageSpeed_Caption;
EveTextSUAverageSpeed.TextAlign = taNone;
EveTextSUAverageSpeed.FontName = 27;
EveTextSUAverageSpeed.Font_Color = 0x0148;
EveTextSUAverageSpeed.FontHandle = 27;
EveTextSUAverageSpeed.Source = -1UL;
EveTextSUAverageSpeed.Active = 1;
EveTextSUAverageSpeed.OnUp = 0;
EveTextSUAverageSpeed.OnDown = 0;
EveTextSUAverageSpeed.OnClick = 0;
EveTextSUAverageSpeed.OnPress = 0;

EveText8.OwnerScreen = &SummaryScreen;
EveText8.Order = 15;
EveText8.Visible = 1;
EveText8.Opacity = 255;
EveText8.Tag = 255;
EveText8.Left = 16;
EveText8.Top = 166;
EveText8.Width = 193;
EveText8.Height = 17;
EveText8.Caption = EveText8_Caption;
EveText8.TextAlign = taNone;
EveText8.FontName = 27;
EveText8.Font_Color = 0x0148;
EveText8.FontHandle = 27;
EveText8.Source = -1UL;
EveText8.Active = 1;
EveText8.OnUp = 0;
EveText8.OnDown = 0;
EveText8.OnClick = 0;
EveText8.OnPress = 0;

BoxRound6.OwnerScreen = &SummaryScreen;
BoxRound6.Order = 16;
BoxRound6.Visible = 0;
BoxRound6.Opacity = 239;
BoxRound6.Tag = 255;
BoxRound6.Left = 28;
BoxRound6.Top = 14;
BoxRound6.Width = 420;
BoxRound6.Height = 241;
BoxRound6.Pen_Width = 1;
BoxRound6.Pen_Color = 0x0000;
BoxRound6.Color = 0xC618;
BoxRound6.Press_Color = 0xE71C;
BoxRound6.Corner_Radius = 3;
BoxRound6.Active = 0;
BoxRound6.OnUp = 0;
BoxRound6.OnDown = 0;
BoxRound6.OnClick = 0;
BoxRound6.OnPress = 0;

EveText9.OwnerScreen = &SummaryScreen;
EveText9.Order = 17;
EveText9.Visible = 0;
EveText9.Opacity = 255;
EveText9.Tag = 255;
EveText9.Left = 42;
EveText9.Top = 20;
EveText9.Width = 57;
EveText9.Height = 21;
EveText9.Caption = EveText9_Caption;
EveText9.TextAlign = taNone;
EveText9.FontName = 28;
EveText9.Font_Color = 0x0148;
EveText9.FontHandle = 28;
EveText9.Source = -1UL;
EveText9.Active = 0;
EveText9.OnUp = 0;
EveText9.OnDown = 0;
EveText9.OnClick = 0;
EveText9.OnPress = 0;

EveText11.OwnerScreen = &SummaryScreen;
EveText11.Order = 18;
EveText11.Visible = 0;
EveText11.Opacity = 255;
EveText11.Tag = 255;
EveText11.Left = 60;
EveText11.Top = 52;
EveText11.Width = 286;
EveText11.Height = 15;
EveText11.Caption = EveText11_Caption;
EveText11.TextAlign = taNone;
EveText11.FontName = 26;
EveText11.Font_Color = 0x0148;
EveText11.FontHandle = 26;
EveText11.Source = -1UL;
EveText11.Active = 0;
EveText11.OnUp = 0;
EveText11.OnDown = 0;
EveText11.OnClick = 0;
EveText11.OnPress = 0;

EveText13.OwnerScreen = &SummaryScreen;
EveText13.Order = 19;
EveText13.Visible = 0;
EveText13.Opacity = 255;
EveText13.Tag = 255;
EveText13.Left = 60;
EveText13.Top = 78;
EveText13.Width = 228;
EveText13.Height = 15;
EveText13.Caption = EveText13_Caption;
EveText13.TextAlign = taNone;
EveText13.FontName = 26;
EveText13.Font_Color = 0x0148;
EveText13.FontHandle = 26;
EveText13.Source = -1UL;
EveText13.Active = 0;
EveText13.OnUp = 0;
```

```
EveText13.OnDown = 0;
EveText13.OnClick = 0;
EveText13.OnPress = 0;

EveText14.OwnerScreen = &SummaryScreen;
EveText14.Order = 20;
EveText14.Visible = 0;
EveText14.Opacity = 255;
EveText14.Tag = 255;
EveText14.Left = 62;
EveText14.Top = 142;
EveText14.Width = 208;
EveText14.Height = 15;
EveText14.Caption = EveText14_Caption;
EveText14.TextAlign = taNone;
EveText14.FontName = 26;
EveText14.Font_Color = 0x0148;
EveText14.FontHandle = 26;
EveText14.Source = -1UL;
EveText14.Active = 0;
EveText14.OnUp = 0;
EveText14.OnDown = 0;
EveText14.OnClick = 0;
EveText14.OnPress = 0;

EveText15.OwnerScreen = &SummaryScreen;
EveText15.Order = 21;
EveText15.Visible = 0;
EveText15.Opacity = 255;
EveText15.Tag = 255;
EveText15.Left = 62;
EveText15.Top = 124;
EveText15.Width = 120;
EveText15.Height = 15;
EveText15.Caption = EveText15_Caption;
EveText15.TextAlign = taNone;
EveText15.FontName = 26;
EveText15.Font_Color = 0x0148;
EveText15.FontHandle = 26;
EveText15.Source = -1UL;
EveText15.Active = 0;
EveText15.OnUp = 0;
EveText15.OnDown = 0;
EveText15.OnClick = 0;
EveText15.OnPress = 0;

EveText16.OwnerScreen = &SummaryScreen;
EveText16.Order = 22;
EveText16.Visible = 0;
EveText16.Opacity = 255;
EveText16.Tag = 255;
EveText16.Left = 62;
EveText16.Top = 160;
EveText16.Width = 93;
EveText16.Height = 15;
EveText16.Caption = EveText16_Caption;
EveText16.TextAlign = taNone;
EveText16.FontName = 26;
EveText16.Font_Color = 0x0148;
EveText16.FontHandle = 26;
EveText16.Source = -1UL;
EveText16.Active = 0;
EveText16.OnUp = 0;
EveText16.OnDown = 0;
EveText16.OnClick = 0;
EveText16.OnPress = 0;

EveText17.OwnerScreen = &SummaryScreen;
EveText17.Order = 23;
EveText17.Visible = 0;
EveText17.Opacity = 255;
EveText17.Tag = 255;
EveText17.Left = 62;
EveText17.Top = 192;
EveText17.Width = 185;
EveText17.Height = 15;
EveText17.Caption = EveText17_Caption;
EveText17.TextAlign = taNone;
EveText17.FontName = 26;
EveText17.Font_Color = 0x0148;
EveText17.FontHandle = 26;
EveText17.Source = -1UL;
EveText17.Active = 0;
EveText17.OnUp = 0;
EveText17.OnDown = 0;
EveText17.OnClick = 0;
EveText17.OnPress = 0;

EveButtonSUBack.OwnerScreen = &SummaryScreen;
EveButtonSUBack.Order = 24;
EveButtonSUBack.Visible = 0;
EveButtonSUBack.Opacity = 255;
EveButtonSUBack.Tag = 255;
EveButtonSUBack.Left = 340;
EveButtonSUBack.Top = 214;
EveButtonSUBack.Width = 96;
EveButtonSUBack.Height = 35;
EveButtonSUBack.Color = 0x03DA;
EveButtonSUBack.Press_Color = 0x7E3F;
EveButtonSUBack.ColorTo = 0x7E3F;
EveButtonSUBack.Press_ColorTo = 0x03DA;
EveButtonSUBack.Caption = EveButtonSUBack_Caption;
EveButtonSUBack.FontName = 27;
EveButtonSUBack.Font_Color = 0xFFFF;
EveButtonSUBack.FontHandle = 27;
EveButtonSUBack.Source = -1UL;
EveButtonSUBack.Flat = 0;
EveButtonSUBack.Active = 0;
EveButtonSUBack.OnUp = 0;
EveButtonSUBack.OnDown = 0;
EveButtonSUBack.OnClick = &EveButtonSUBack_OnClick;
EveButtonSUBack.OnPress = 0;

EveButtonSUBack_OnClick.Action =
EveButtonSUBackOnClick;
EveButtonSUBack_OnClick.Sound.SndAct =
VTFT_SNDACT_NONE;
EveButtonSUBack_OnClick.Sound.Effect =
_FT800_SOUND_XYLOPHONE;
EveButtonSUBack_OnClick.Sound.Pitch =
_FT800_SOUND_PITCH_A5;
EveButtonSUBack_OnClick.Sound.Volume = 255;

}

void Init_MCU() {
PLLFBD = 68;
CLKDIV = 0x0000;

ANSELA = 0x00;
ANSELB = 0x00;
}

void InitVTFTStack() {
union {
TFT800PWM PWM;
TFT800GPIO GPIO;
TFT800Audio Audio;
TFT800Sound Sound;
TFT800Touch Touch;
```

```
 TFT800Display Display;
 TFT800Interrupt Interrupt;
 TFT800TouchTransform TTransform;
 } cfg;

 Init_MCU();

 SPI1_Init_Advanced(_SPI_MASTER,
 _SPI_8_BIT,
 _SPI_PRESCALE_SEC_2,
 _SPI_PRESCALE_PRI_4,
 _SPI_SS_DISABLE,
 _SPI_DATA_SAMPLE_MIDDLE,
 _SPI_CLK_IDLE_LOW,
 _SPI_IDLE_2_ACTIVE);




 FT800_Init();



FT800_Core_ClockSource(_FT800_CLK_SOURCE_EXTERNAL
);
 FT800_Core_ClockPLL(_FT800_CLK_PLL_48MHz);


 cfg.Display = VTFT_FT800_CONFIG_DISPLAY;
 FT800_Display_SetConfig(&cfg.Display);

 cfg.PWM = VTFT_FT800_CONFIG_PWM;
 FT800_PWM_SetConfig(&cfg.PWM);

 cfg.GPIO = VTFT_FT800_CONFIG_GPIO;
 FT800_GPIO_SetConfig(&cfg.GPIO);

 cfg.Audio = VTFT_FT800_CONFIG_AUDIO;
 FT800_Audio_SetConfig(&cfg.Audio);

 cfg.Sound = VTFT_FT800_CONFIG_SOUND;
 FT800_Sound_SetConfig(&cfg.Sound);

 cfg.Interrupt = VTFT_FT800_CONFIG_INTERRUPT;
 FT800_Interrupt_SetConfig(&cfg.Interrupt);

 cfg.Touch = VTFT_FT800_CONFIG_TOUCH;
 FT800_Touch_SetConfig(&cfg.Touch);

 cfg.TTransform =
VTFT_FT800_CONFIG_TOUCHTRANSFORM;
 FT800_TouchTransform_SetConfig(&cfg.TTransform);


 InitObjects();


 DrawScreen(&SplashScreen);
}
```