

Design Specification

- Szczegóły techniczne dla rozwiązań opisanych w FS

**Innowacyjny system do kompleksowej ozonoterapii
stomatologicznej oraz dekontaminacji unitu przy wykorzystaniu
wysokiego stężenia ozonu w różnych formach**

Author: Paweł Barc

Date: 20/07/2016

Version: 1.0

Document Control

Document location

Location

Author

Position	Name	Contact no
Konstruktor Elektronik	Paweł Barc	

Revision history

Version	Issue date	Author/editor	Description/Summary of changes
1.0	22/07/2016	Grzegorz Karliński	

Reviewed by

Version	Issue date	Name	Position	Review date
1.0	25/07/2016	Jacek Olszewski	Konstruktor	16/08/2016

Approvals

Version	Issue date	Name	Position	Approval date
1.0	16/08/2016	K. Wójtowicz	Dyrektor R&D	26/08/2016

Related documents

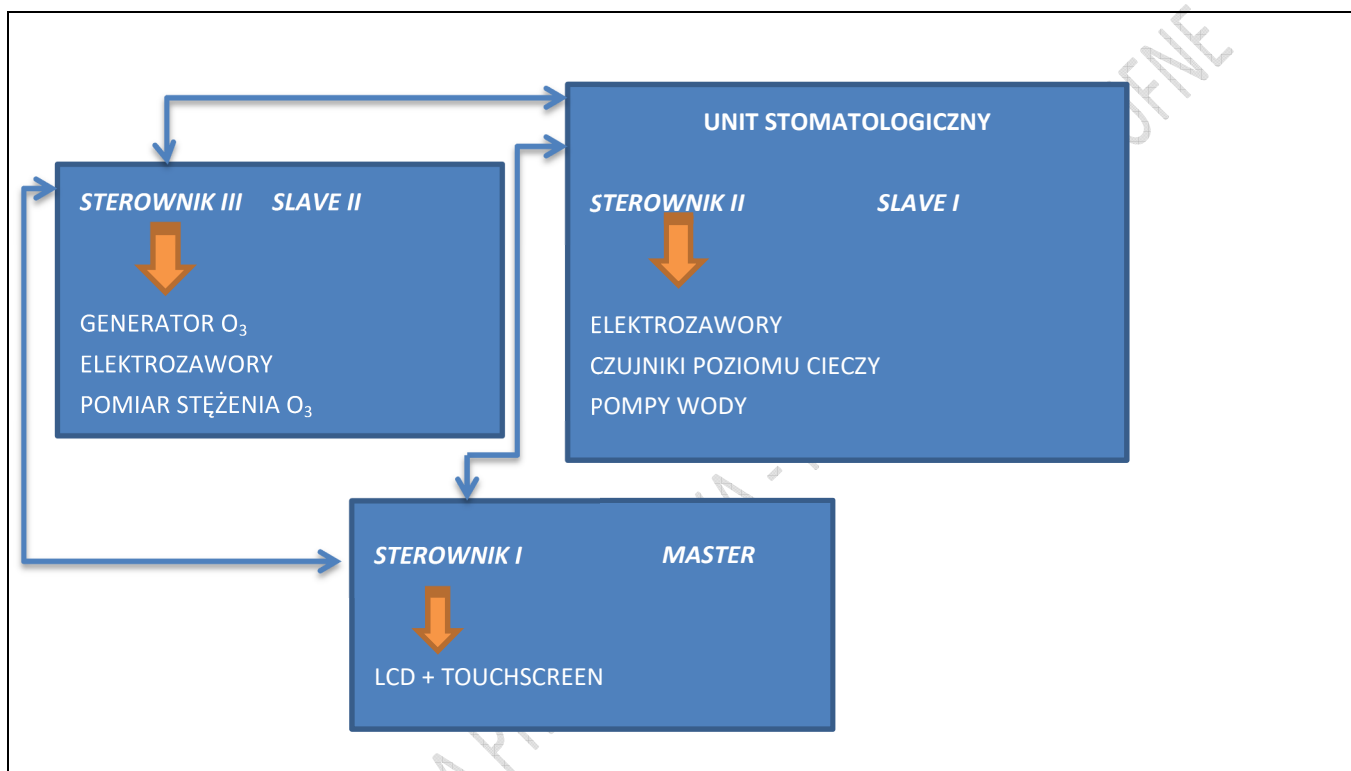
Document	Location

1.1. Introduction

1.1.1. Purpose

Dokument zawiera szczegóły techniczne dla rozwiązań opisanych w FS

1.2. System Architecture and Architecture Design



1.2.1. Hardware Architecture

Platforma sprzętowa:

- mikrokontroler jednoukładowy PIC32MX360F512L
- pamięć AT45DB642

Narzędzia programistyczne:

- MPLAB v8.60 (lub wyższa)
- MPLAB C32 C Compiler v1.04 (lub wyższa)

Interfejs użytkownika:

- wyświetlacz kolorowy z ekranem dotykowym, 2.5 cala , 320x240
- dwufunkcyjny przycisk nożny

Sterowania pozostałe:

- pompa ssąca
- sterowanie włączeniem generatorem ozonu
- rezystorowe, przekaźnikowe ustawianie stężeń,
- czujniki stężenia ozonu
- sterowany wentylator generatora ozonu
- pięć elektrozaworów

-dwufunkcyjny przycisk nożny

Informacje/klawisze wyświetlane na ekranie głównym

- 4 klawisze wyboru zabiegu
- ząbek
- kanał
- butelka
- szklanka
- klawisz konfiguracji -wybór języka aparatu
- ikona informująca o pracy generatora ozonu
- informacja o stanie wody ozonowanej w zbiorniku, ikona, plus licznik czasu „przydatności”

Działanie:

- możliwości ustalenia czasu zabiegu i stężenia dla każdego zabiegu
- automatyczne rozpoczęcie automatycznych procedur związanych ze zbyt wysokim stężeniem ozonu
- informacja o upływie czasu i wybranym stężeniu
- wybór stężenia z predefiniowanych poziomów
- tryb serwisowy pracy umożliwiający:
 - testy poszczególnych podzespołów aparatu
 - kalibrację ekranu dotykowego

1.2.2. Software Architecture

Na oprogramowaniu, projekt w środowisku MPLAB, składają się następujące pliki.

- | | |
|----------------------|--|
| -AMPIRE_8bit.c | -obsługa wyświetlacza graficznego |
| -AT45.c | -obsługa zewnętrznej pamięci flash |
| -NANO.c | -główny plik programu |
| -bmp_ext_nano.c | -obsługa bitmap w pamięci zewnętrznej |
| -bmp_ikony_nano.c | -bitmapy ikon w pamięci wewnętrznej |
| -bmp_klawisze_nano.c | -bitmapy klawiszy w pamięci wewnętrznej |
| -Fonts.c | -fonty biblioteczne |
| -fonty_Arial.c | -pliki z wykorzystywanymi fontami |
| -fonty_engravers.c | -pliki z wykorzystywanymi fontami |
| -fonty_tahoma.c | -pliki z wykorzystywanymi fontami |
| -funkcje_hardware.c | -funkcje obsługi modułów sprzętowych |
| -funkcje_soft.c | -funkcje pomocnicze wykorzystywane przez inne moduły |
| -rs232.c | -funkcje obsługi komunikacji szeregowej przez RS-a |
| -TouchScreen.c | -funkcje obsługi ekranu panelu dotykowego |
|
 | |
| -GOL.c | -funkcje graficzne, biblioteka moduł główny |
| -GOLFontDefault.c | -pomocnicze funkcje graficzne biblioteka |
| -Picture.c | -pomocnicze funkcje graficzne biblioteka |
| -Primitive.c | -pomocnicze funkcje graficzne biblioteka |
| -Slider.c | -pomocnicze funkcje graficzne biblioteka |
| -StaticText.c | -pomocnicze funkcje graficzne biblioteka |
| -Window.c | -pomocnicze funkcje graficzne biblioteka |

1.3. Detailed Software

Sposób implementacji poszczególnych jednostek programowych:

- ogólna konfiguracja procesora i programu
- Plik NANO.c

- KONFIGURACJA PROCESORA

```
#pragma config FPLLDIV = DIV_2, FPLLMUL = MUL_20, FPLLODIV = DIV_1, FWDTEN = OFF, FSOSCEN = OFF,
WDTPS = PS512
#pragma config POSCMOD = XT, FNOSC = PRIPLL, FPBDIV = DIV_1
#pragma config CP = OFF, BWP = OFF, PWP = OFF
```

- Funkcja `int main()`

```
SYSTEMConfigPerformance(GetSystemClock());
mOSCSetPBDIV(OSC_PB_DIV_1);
DDPCONbits.JTAGEN = 0;           //pozwala na używanie bitów RB10 do RB13,
GOL_MSG msg;                     // definicja - GOL message structure to interact with GOL
GOLInit();                       // inicjalizacja biblioteki graficznej
Przerwania_Init ();
AT45_Init();
LCD_LED_INIT;
LCD_LED_ON;
UstawJasnosc(JASNOSC_LED_MIN); //ekran ciemny
TouchInit();
Init_Wejsc_Wyjsc();
```

- obsługa wyświetlacza

- Plik `AMPIRE_8bit.c`

- funkcje

```
ResetDevice(void)
PutPixel(SHORT x, SHORT y)
Bar(SHORT left, SHORT top, SHORT right, SHORT bottom)
ClearDevice(void)
PutImage(SHORT left, SHORT top, void* bitmap, BYTE stretch)
PutImageExt(EXTDATA* memory, SHORT left, SHORT top)
```

- obsługa ekranu dotykowego

- Plik `TouchScreen.c`

- funkcje

```
void TouchInit(void)
void __T3_ISR_T3Interrupt(void){
void TouchGetMsg(GOL_MSG *pMsg)
void TouchCalibration(void)
```

- obsługa zewnętrznej pamięci grafiki AT45...

- Plik `AT45.c`

- funkcje

```
void main_write_through_buffer1(uint16_t adres,unsigned char *data,unsigned int nbytes )
void main_array_read(uint32_t adres,unsigned char *data,unsigned int nbytes)
char status_read(void)
```

- obsługa wejść/wyjść binarnych

- Plik `NANO.c`

- funkcje

```
void Init_Wejsc_Wyjsc(void)
void OdczytStanuWejsc()
void SterowanieWy(void)
```

-implementacja ekranów zabiegów

Plik NANO.c

-funkcje

```
void ZmianyEkranuButla()
void ZmianyEkranuKanal()
void ZmianyEkranuMain()
void ZmianyEkranuSzlanka()
void ZmianyEkranuZombek()
void CreateEkranButla()
void CreateEkranKanal()
void CreateEkranMain()
void CreateEkranSzlanka()
void CreateEkranZombek()
```

-komunikacja z użytkownikiem

Plik NANO.c

-funkcje

```
WORD GOLDrawCallback()
WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER* pObj, GOL_MSG* pMsg)
void DisplayMinSek(SHORT x, SHORT y)
void DisplayCzas_9_59(char min,char dsek,char jsek, int kolor)
void DisplayCzasButli(char dmin, char jmin, char dsek, char jsek, int kolor)
```

-implementacja komunikatów o stanach i ikon

Plik NANO.c

-funkcje

```
void CreateEkranOkienko(void)
WORD GOLDrawCallback()
if(rysuj_ikony){
    rysuj_ikony=0;
    rysuj_piorun=1;
    rysuj_butelka=1;
    rysuj_czas_butli=1;
}
if(rysuj_piorun){
    rysuj_piorun=0;
    if(stan_O3==STAN_ON) {
        PutImage(160,3, (void*)&ico_piorun_luz8,IMAGE_NORMAL); }
    if(stan_O3==STAN_OFF){
        PutImage(160,3, (void*)&ico_piorun_tlo8,IMAGE_NORMAL); }
}
if(rysuj_butelka){
    rysuj_butelka=0;
    if(stan_butli==BUTLA_O3_BRAK){
        PutImage(286,4, (void*)&ico_butelka_szara8,IMAGE_NORMAL); }
    if(stan_butli==BUTLA_O3_OK) {
        PutImage(286,4, (void*)&ico_butelka8,IMAGE_NORMAL); }
}
```

-implementacja funkcji serwisowych/testowych/kalibracyjnych

Plik NANO.c

-funkcje

```
void CreateEkranASerwis()
void CreateEkranATesty()
```

```

void ImportDaty(char typ)
void KalibracjaToucha(void)

/*****
*          .C          *
*      plik main      *
*****/
#include <plib.h>
#include <stdio.h>

#include "NANO.h"
//=====
//UWAGA 20 - pinowe złącze I/O, trzecia wspólnego wersja pcb
//zmiana przyporządkowania zaworów do funkcji, ze względu na zawory zewnętrzne
//zamiana V3 / V4
// wersja TETOWA EMC zmiana L:1083
//===== KONFIGURACJA PROCESORA
=====

#pragma config FPLLDIV = DIV_2, FPLLMUL = MUL_20, FPLLODIV = DIV_1, FWDTEN = OFF, FSOSCEN = OFF, WDTPS =
PS512
#pragma config POSCMOD = XT, FNOSC = PRIPLL, FPBDIV = DIV_1
#pragma config CP = OFF, BWP = OFF, PWP = OFF

//===== prototypy funkcji =====
//===== testowe =====
//----
void CreateEkranASerwis();
WORD MsgEkranZero(WORD objMsg, OBJ_HEADER* pObjj);
//----
//void CreateEkranTestMP3();
//WORD MsgEkranTestMP3(WORD objMsg, OBJ_HEADER* pObjj);
//----
void CreateEkranTesty();
void CreateEkranTestStatyczny();
void CreateEkranATesty();
WORD MsgEkranTesty(WORD objMsg, OBJ_HEADER* pObjj);

void KalibracjaToucha();
//===== testowe - end =====
//prototypy funkcji pomocniczych
void AktywacjaMP3(void);
void OutTextXJustedY(SHORT x,SHORT y, char just, XCHAR * my_string, void * font);
void Beep();
char AnyKey(void);
void DisplayTup();
void DisplayTupDot();
void DispPresuryName();
void DisplayPresury();
void DisplayStanOut();
void DisplayStanInt();
void DisplayInputy();
void DisplaySlidery();
void DisplayZmianyT();
void DisplayZmianySTE(int poz);
//void GetZdarzenieInput();
//void ProcesZdarzenieInput();
void DisplayCzasSekundy(char czas, int kolor);
void DisplayCzas_9_59(char min,char dsek,char jsek, int kolor);

```

```

void DisplayCzasButli(char dmin, char jmin, char dsek, char jsek, int kolor);
void DisplayMinutyOzon(char czas, int kolor);
void DisplayDawka(char d1, char d2, char d3, int kolor);
void DisplayDawkaX(WORD dana,int kolor);
void SterowanieWy(void);
void SterowanieSerwisowe(void);
void UstawStezenieRobocze(WORD stez);
void ZaworyOff(void);
void OdczytStanuWejsc();
void NadzorStanuButli();
void ImportBitmap(void);
void ImportDaty(char typ);
void ZapisKonfiguracji(void);
void OdczytKonfiguracji(void);
void UstawJasnosc(WORD jasnosc);
void DrawSuwak(int kolor,SHORT x1,SHORT y1,SHORT x2,SHORT y2,WORD zakres,WORD poz);
void DispSuwakV(SHORT x,SHORT y1,SHORT y2,WORD zakres,WORD poz);
void DiodyRGB_On(void);
void DiodyRGB_Off(void);
void UstawVolume(char volume);

//prototypy funkcji stanowych
void PrzeliczenieCzasu(WORD czas);
void PokazCzas(SHORT x, SHORT y, char h, char min);
void DisplayMinSek(SHORT x, SHORT y);
//=====
void CreateEkranStartowy();
void CzekajEkranStartowy();
WORD MsgEkranStartowy(WORD objMsg, OBJ_HEADER* pObj);
//=====ekran główny=====
void CreateEkranMain();
WORD MsgEkranMain(WORD objMsg, OBJ_HEADER* pObj);
void ZmianyEkranuMain();

void CreateEkranZombek();
WORD MsgEkranuZombek(WORD objMsg, OBJ_HEADER* pObj);
void ZmianyEkranuZombek();

void CreateEkranKanal();
WORD MsgEkranuKanal(WORD objMsg, OBJ_HEADER* pObj);
void ZmianyEkranuKanal();

void CreateEkranButla();
WORD MsgEkranuButla(WORD objMsg, OBJ_HEADER* pObj);
void ZmianyEkranuButla();

void CreateEkranSzkłanka();
WORD MsgEkranuSzkłanka(WORD objMsg, OBJ_HEADER* pObj);
void ZmianyEkranuSzkłanka(void);

void CreateEkranOkienko(void);
WORD MsgEkranuOkienko(WORD objMsg, OBJ_HEADER* pObj);
void ZmianyEkranuOkienko(void);

void CreateEkranConfig();
WORD MsgEkranConfig(WORD objMsg, OBJ_HEADER* pObj);

void CreateEkranInfo();
void ZmianyEkranuInfo();

```



```
//=====
//          Deklaracja ID obiektów
//=====

#define ID_BUTTON_A      1
#define ID_BUTTON_B      2
#define ID_BUTTON_C      3

#define ID_BUTTON_T1      4
#define ID_BUTTON_T2      5
#define ID_BUTTON_T3      6
#define ID_BUTTON_T4      7
#define ID_BUTTON_OKNO    8

#define ID_WINDOW1      10

#define ID_BUTTON1      11
#define ID_BUTTON2      12
#define ID_BUTTON3      13
#define ID_BUTTON4      14
#define ID_BUTTON5      15
#define ID_BUTTON6      16
#define ID_BUTTON7      17
#define ID_BUTTON8      18
#define ID_BUTTON9      19

#define ID_BUTTON_V1     20
#define ID_BUTTON_V2     21
#define ID_BUTTON_V3     22
#define ID_BUTTON_V4     23
#define ID_BUTTON_V5     24
#define ID_BUTTON_V6     25
#define ID_BUTTON_INFO   26
#define ID_BUTTON_TPLUS   27
#define ID_BUTTON_TMINUS  28
#define ID_BUTTON_POMPA1  29
#define ID_BUTTON_POMPA2  30
#define ID_BUTTON_O3      31
#define ID_BUTTON_FAN     32

#define ID_BUTTON_BACK    33
#define ID_BUTTON_PK1     34
#define ID_BUTTON_PK2     35
#define ID_BUTTON_RED     36
#define ID_BUTTON_GREEN   37
#define ID_BUTTON_BLUE    38

#define ID_BUTTON_STOP    40
#define ID_BUTTON_START   41
#define ID_BUTTON_MAIN    42
#define ID_BUTTON_SET     43
#define ID_BUTTON_CZAS    44
#define ID_BUTTON_OZON    45

#define ID_SLIDER1        51
#define ID_SLIDER2        52
```

```

#define ID_PROGRESSBAR1          60
//=====
//zmienne dostępu do bitmap w pamięci zewnętrznej
//=====
extern BITMAP_EXTERNAL ekr_start;
//extern BITMAP_EXTERNAL ekr_konfig_demo;
extern BITMAP_EXTERNAL ekr_konfig;
extern BITMAP_EXTERNAL ekr_main_AN;
extern BITMAP_EXTERNAL ekr_main_PL;
extern BITMAP_EXTERNAL ekr_zabieg_AN;
extern BITMAP_EXTERNAL ekr_zabieg_PL;

extern BITMAP_EXTERNAL flagi_blue;
extern BITMAP_EXTERNAL klik;
extern BITMAP_EXTERNAL klementyna;
extern BITMAP_EXTERNAL kasia;
extern BITMAP_EXTERNAL flagi;

//=====
//zmienne dostępu do dźwięków w pamięci zewnętrznej
//=====

//=====
//zmienne dostępu do bitmap klawiszy, w pamięci internal flash
//=====
// w pliku bmp_int_nano.h

//=====
//zmienne dostępu do czcionek, w pamięci internal flash
//=====
//plik fonty_engravers.c
extern const FONT_FLASH Engravers_8;           //cyfry
//extern const FONT_FLASH Engravers_12;        //cyfry
extern const FONT_FLASH Engravers_16;          //cyfry -pokazywanie upływu czasu zabiegu
//extern const FONT_FLASH Engravers_20;        //cyfry
//extern const FONT_FLASH Engravers_24;        //cyfry
//extern const FONT_FLASH Engravers_28;        //cyfry

//plik fonty_tahoma_cyf_80
extern const FONT_FLASH Tahoma_cyfry_80;       //cyfry -ustawianie czasu
//plik GOLFontDefault.c
extern const FONT_FLASH GOLFontDefault; // default GOL font
//plik Fonts.c
extern const FONT_FLASH GOLMediumFont;          // medium font
extern const FONT_FLASH GOLSmallFont;          // small font
//plik fonty_internal.c
extern const FONT_FLASH Verdana_standard_10;
extern const FONT_FLASH Verdana_standard_12;
//extern const FONT_FLASH Verdana_standard_8;
//extern const FONT_FLASH Verdana;
//extern const FONT_FLASH Verdana_10;
//extern const FONT_FLASH Verdana_12;
//extern const FONT_FLASH Verdana_polskie_8;
//plik fonty_Arial.c
extern const FONT_FLASH Arial_bold_6;
extern const FONT_FLASH Arial_6;

```

```

//extern const FONT_FLASH V_cyfry_10;
//extern const FONT_FLASH V_cyfry_11;
//extern const FONT_FLASH V_cyfry_12;
//plik fonty_tahoma_pl_6.c
extern const FONT_FLASH Tahoma_pl_6;
extern const FONT_FLASH Tahoma_Bold_pl_6;

//extern const FONT_FLASH Verdana_B7;
//extern const FONT_FLASH Verdana_B6;
//===== DEFINICJE ZMIENNYCH GLOBALNYCH =====
//*****
PROGRESSBAR* pProgressBar; // pointer to progress bar object
SLIDER* pSlider, pSlider1, pSlider2;
BUTTON* pButon_A, pButon_B, pButon_C;

GOL_SCHEME* pBarScheme;
GOL_SCHEME* BtnScheme;
GOL_SCHEME* BtnOffScheme;
GOL_SCHEME* altScheme;
GOL_SCHEME* ValvScheme;
GOL_SCHEME* OutOnScheme;
GOL_SCHEME* OutOffScheme;
GOL_SCHEME* ProcScheme; // przyciski

//*****
typedef enum {
    OZON_OFF = 0, //wyłączony
    OZON_START, //włączanie
    OZON_ON, //włączony
    OZON_STOP, //wyłączanie
} FAZY_OZONU;
FAZY_OZONU faza_generacji = OZON_OFF; // aktualny status generowania ozonu

typedef enum {
    BRAK_EVENTA = 0,
    NOGAL1_PRESS, NOGAL1_RELEASE,
    NOGAL2_PRESS, NOGAL2_RELEASE,
} INPUT_EVENT;
INPUT_EVENT input_event = BRAK_EVENTA; //odczytane zdarzenia z wejść binarnych

typedef enum {
    BUTLA_O3_BRAK=0,
    BUTLA_O3_GRANICA,
    BUTLA_O3_OK,
} STAN_BUTLI;
STAN_BUTLI stan_butli = BUTLA_O3_BRAK;

//zmienna - indeks tablicy konfiguracji
char konfiguracja_aparatu[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

typ_Jezyk jezyk=ANGIELSKI; //POLSKI; //zmienna wyboru języka /

char jasnoc_led=JASNOSC_MAX;
char disp_jasnoc_led=JASNOSC_MAX; //wyświetlana = poprzednia jasność
char volume_tekst=0; //głośność komunikatów
//char disp_volume_tekst=0; //poprzednia głośność komunikatów
char volume_dzwiek=23; //głośność sygnałów dźwiękowych, zakres 1-40

```

```

char disp_volume_dzwiek=1;                                     //poprzednia głośność sygnałów dźwiękowych

//flagi przerysowywania grafiki
char rys_jasnosc_led=0;

char rys_jezyk_flaga=0;
//char rys_volume=0;

char spi_ch2_busy;                                             //flaga zajętości kanału 2 spi przez pamięć AT45
char mem_use;                                                  //flaga pamięć w użyciu - nie tykać

//*****
STANY_EKRANU ekran = CREATE_STARTOWY; // ... , aktualny stan aparatu = stanowi ekranu
//deklaracje stanów ekranu ==> pracy aparatu w pliku *.h
//*****
unsigned int tst_1=0, tst_2=0, tst_3=0;
unsigned int t_touch=0, pomiar_abs=0, pomiar_diff=0;
unsigned int wartosc_t=0, wartosc_s=0;                          //zmienne czasu i stężenia

//----- zmienne obsługi wejść -----
char nogal_blue=0;
char nogal_yellow=0;

//----- zmienne obsługi wyjść -----

char stan_V1=0, stan_V2=0, stan_V3=0, stan_V4=0, stan_V5=0, stan_V6=0;
char stan_pompy=0;
char stan_pompa1=0;
char stan_pompa2=0;
char stan_fan=0;
char stan_O3=0;
char stan_PK1=0;
char stan_PK2=0;

char stan_red=0;
char stan_green=0;
char stan_blue=0;

char stan_relayow=0;                                           //flaga - przekaźniki poziomu aktywne
char stan_generatora=0;                                       //flaga - generator ozonu aktywny
unsigned int poziom_rel=0;                                     //poziom "przekaźnikowy" ozonu
//stany wyjść - SERWISOWE
char stan_LE1=0;
char stan_LE2=0;
char stan_LE3=0;
char stan_PECs=0;
//-----
char repeat_flag=0;                                           // flaga autorepetycji bieżącego klawisza
char flaga=0;
//char rysuj_ste=0;                                           // flaga odświeżenia zmiany ste..
WORD stezenie=30;

WORD o3krwi_dawka=0;                                          // wybrana dawka dla ozonowania krwi - ostatnio..
WORD dawka;                                                  // pomocnicza do ekranu zmiany dawki

```

```

WORD o3krwi_czas_zabiegu=0;           // czas zabiegu wyliczony z dawki
WORD dawka_podana=0;                  // pomocnicza
char zab_dawka1=0, zab_dawka2=2, zab_dawka3=0;
unsigned int zab_czas=0;
char pauza_zabiegu=0;                // flaga informująca o pauzie w zabiegu

WORD czas_pauza_zabiegu=0;           // zapamiętany pozostały czas zabiegu podczas pauzy, sekundy
//----
WORD ust_czas_zabiegu=0;              // ustawialny czas trwania zabiegu - sekundy,
WORD oliwa_czas_zabiegu=3;           // czas zabiegu -minuty, do zapamiętania nastawy
//----
WORD banka_czas_zabiegu=5;           // tu jako poczatkowy - minuty, do zapamiętania nastawy
WORD banka_stezenie=5;               // podstawiam minimalne
WORD banka_nieszczelna=0;           // licznik czasu nieszczelności bańki, gdy za długo to STOP
//----
WORD rekaw_czas_zabiegu=5;           // tu jako poczatkowy - minuty, do zapamiętania nastawy
WORD rekaw_stezenie=5;               // podstawiam minimalne
//----
WORD jamy_czas_zabiegu=3;            // minuty
WORD jamy_stezenie=5;                // podstawiam minimalne
//----
WORD plyny_czas_zabiegu=3;           // minuty
WORD plyny_stezenie=5;               // podstawiam minimalne
//----
WORD ostrzyk_czas_zabiegu=3;         // minuty
WORD ostrzyk_stezenie=5;             // podstawiam minimalne

char start_odwilgania=0;              //flaga sterowania usuwaniem wilgoci

char czas_jh=0, czas_dm=0, czas_jm=0, czas_ds=0, czas_js=0; //cyfry czasu jh : dm,jm : ds,js
//-----
char czas_ozon_max, czas_ozon_min;    //czasy dla zabiegów Płyn, Rekaw, Jamy w minutach , oraz ich ograniczenia,
char czas_ozon=0;                     //ustawiony w danej chwili czas_ozon

char disp_czas_sekundy;                //zapamiętany wyświetlony czas
char disp_czas_ozon;                   //wyświetlany w danej chwili czas_ozon

char rys_dawka=0;
char disp_dawka1;                      //wyświetlana cyfra1 dawki
char disp_dawka2;
char disp_dawka3;

//===== na pewno użyte =====
//-----
char tryb_serwisowy=0;                  //program w trybie serwisowym
char testowy_display=0;                 //wyświetlanie stanów we/wy na górnym pasku tła
//-----
char flaga_zmiany_okna=0;              //
char zmien_strone_graf=0;
char strona_graf=1;
char faza_animacji=0;                  //synchronizator faz animacji, zerowane przy CreateEkran...
char animacje_aktywne=0;               //ON-OFF animacji ogólnie
char animacja_O3=0;                   //zapamiętuje czasowo - stan_O3
char animacja_butli=BUTLA_O3_BRAK;

//===== nowe / nowo użyte =====
//flagi i zmienne "graficzne" , do odświeżania elementów ekranów

```

```

WORD czas_sekundy_max, czas_sekundy_min;
WORD czas_sekundy=0;           //ustawiony w danej chwili czas
char rysuj_czas_sekundy=0;      //flaga przerysowania
char rysuj_czas_599=0;          //flaga przerysowania
char rysuj_czas_butli=0;        //flaga przerysowania
char rysuj_ikony=0;
char rysuj_butelka=0;
char rysuj_piorun=0;
WORD stezenie_ozonu=1;          //do wyboru stężenia A=1, B=2, C=3 w zabiegach zębowych, zapamiętywane
po zmianie
WORD stezenie_robocze=1;        //wykonawcze
WORD czas_zabiegu=CZAS_ZABIEGU_SET; //ustawiony czas zabiegu, sekundy
char blokada=0;                 //flaga blokady nogala_yellow po skończeniu zabiegu
char stan_zabiegu=0;            //stany
char stan_pauzy=0;              //trzy stany, brak, ustawianie i trwanie
WORD czas_pauza=0;              //zapamiętywany przy wejściu w pauzę, czas pozostały do końca
zabiegu
WORD czas_ozonowania_butli=CZAS_BUTLI;
char errorokno=0;
char stan_mp3=0;                //stan inicjalizacji i status VS1011, 0=nie działa

unsigned int offset_adresu=0;    //offset dla odczytu bitmap z pamięci
unsigned int main_cykl=0;        //licznik petli main

//=====

void WymiarFonta(void){
    LiczbaToText(GetTextHeight((void*)&Verdana_standard_10),2,&text_bufor[0]);
    SetColor(WHITE);
    SetFont((void*)&Tahoma_cyfry_80);
    OutTextXJustedY(0,90,CENTRE_JUSTED,&text_bufor[0],(void*)&Tahoma_cyfry_80);
}
//*****schemat kolorow guzika*****
#define DARKGREEN    RGB565CONVERT( 0, 100,  0)
#define PALEGREEN    RGB565CONVERT(152, 251, 152)
#define RED4          RGB565CONVERT(139,  0,  0)
#define FIREBRICK1    RGB565CONVERT(255, 48, 48)
#define MYYELLOW      RGB565CONVERT(0xFF, 0xC0, 0)
#define TLO_ATO3      RGB565CONVERT(75, 155, 0)
//RGB565CONVERT(133, 194, 38)
#define TLO_NR2        RGB565CONVERT(180, 180, 180)
//RGB565CONVERT(133, 194, 38)
#define TLO_NR3        RGB565CONVERT(75, 75, 75)
#define SZARA_CYFRA    RGB565CONVERT(125, 125, 125)
#define GREGWHITE      RGB565CONVERT(251, 251, 251)
#define TLO_GREEN      RGB565CONVERT(136, 217, 31)
#define GREGGREY       RGB565CONVERT(234, 234, 234)

int main(void){

    BtnScheme = GOLCreateScheme();
    BtnScheme->Color0 = MYYELLOW;           //RGB565CONVERT(0xFF, 0xC0, 0);
    BtnScheme->Color1 = RGB565CONVERT(0xFF, 0xC0, 0);
    BtnScheme->EmbossDkColor = RGB565CONVERT(55,55,55);
    BtnScheme->EmbossLtColor = RGB565CONVERT(0xAC, 0xB5, 0xB8);
    BtnScheme->ColorDisabled = BLACK;
    BtnScheme->TextColor1 = RGB565CONVERT(0x5C, 0x8E, 0xFF);
    BtnScheme->TextColor0 = DARKGREEN;      //RGB565CONVERT(0xFF, 0xCB, 0x3C);
    BtnScheme->TextColorDisabled = BLACK;

```



```

BtnOffScheme = GOLCreateScheme();
    BtnOffScheme->Color0 = LIGHTCYAN;           //RGB565CONVERT(0xFF, 0xC0, 0);
    BtnOffScheme->Color1 = LIGHTCYAN;           //RGB565CONVERT(0xFF, 0xC0, 0);
    BtnOffScheme->EmbossDkColor = RGB565CONVERT(55,55,55);
    BtnOffScheme->EmbossLtColor = RGB565CONVERT(0xAC, 0xB5, 0xB8);
    BtnOffScheme->ColorDisabled = BLACK;
    BtnOffScheme->TextColor1 = RGB565CONVERT(0x5C, 0x8E, 0xFF);
    BtnOffScheme->TextColor0 = DARKGREEN;       //RGB565CONVERT(0xFF, 0xCB, 0x3C);
    BtnOffScheme->TextColorDisabled = BLACK;

OutOnScheme = GOLCreateScheme();
    OutOnScheme->pFont = (void*)&GOLMediumFont;
    OutOnScheme->Color0 = RED;
    OutOnScheme->Color1 = LIGHTBLUE;
    OutOnScheme->EmbossDkColor = RGB565CONVERT(55,55,55);
    OutOnScheme->EmbossLtColor = RGB565CONVERT(0xAC, 0xB5, 0xB8);
    OutOnScheme->ColorDisabled = BLACK;
    OutOnScheme->TextColor1 = RED;               //RGB565CONVERT(0x5C, 0x8E, 0xFF);
    OutOnScheme->TextColor0 = BRIGHTBLUE;       //RGB565CONVERT(0xFF, 0xCB, 0x3C);
    OutOnScheme->TextColorDisabled = BLACK;

OutOffScheme = GOLCreateScheme();
    OutOffScheme->pFont = (void*)&GOLMediumFont;
    OutOffScheme->Color0 = GREEN;
    OutOffScheme->Color1 = LIGHTBLUE;
    OutOffScheme->EmbossDkColor = RGB565CONVERT(55,55,55);
    OutOffScheme->EmbossLtColor = RGB565CONVERT(0xAC, 0xB5, 0xB8);
    OutOffScheme->ColorDisabled = BLACK;
    OutOffScheme->TextColor1 = RED;               //RGB565CONVERT(0x5C, 0x8E, 0xFF);
    OutOffScheme->TextColor0 = BRIGHTBLUE;       //RGB565CONVERT(0xFF, 0xCB, 0x3C);
    OutOffScheme->TextColorDisabled = BLACK;

altScheme = GOLCreateScheme();                // create alternative 1 style scheme
    altScheme->Color0 = RGB565CONVERT(0x4C, 0x8E, 0xFF);
    altScheme->Color1 = RGB565CONVERT(0xFF, 0xBB, 0x4C);
    altScheme->EmbossDkColor = RGB565CONVERT(0x1E, 0x00, 0xE5);
    altScheme->EmbossLtColor = RGB565CONVERT(0xA9, 0xDB, 0xEF);
    altScheme->ColorDisabled = RGB565CONVERT(0xD4, 0xE1, 0xF7);
    altScheme->TextColor1 = BRIGHTBLUE;
    altScheme->TextColor0 = RGB565CONVERT(0xFF, 0xBB, 0x4C);
    altScheme->TextColorDisabled = RGB565CONVERT(0xB8, 0xB9, 0xBC);

ProcScheme = GOLCreateScheme();                // schemat przycisków
    ProcScheme->pFont = (void*)&Tahoma_pl_6;      //(void*)&Verdana_standard_10;
    ProcScheme->Color0 = RGB565CONVERT(208, 209, 210); //RGB565CONVERT(222, 222, 222);
    ProcScheme->Color1 = RGB565CONVERT(208, 209, 210); //RGB565CONVERT(77, 77, 77);
    ProcScheme->EmbossDkColor = TLO_NR3; //RGB565CONVERT(208, 209, 210); //TLO_ATO3;
    //RGB565CONVERT(208, 209, 210);
    ProcScheme->EmbossLtColor = TLO_NR2; //WHITE; //RGB565CONVERT(0xA9, 0xDB, 0xEF);
    ProcScheme->ColorDisabled = RGB565CONVERT(0xD4, 0xE1, 0xF7);
    ProcScheme->TextColor1 = RGB565CONVERT(208, 209, 210); //TLO_ATO3; //RGB565CONVERT(133,
194, 38); //RGB565CONVERT(0xBB, 0xBB, 0x4C);
    ProcScheme->TextColor0 = BLACK;               //RGB565CONVERT(150, 150, 150); //szary klawisz
nie wybranego stężenia
    ProcScheme->TextColorDisabled = RGB565CONVERT(0xB8, 0xB9, 0xBC);

pBarScheme = GOLCreateScheme();
    pBarScheme->pFont = (void*)&GOLFontDefault;    //&GOLMediumFont;

```

```

pBarScheme->Color0 = RGB565CONVERT(0, 0, 0);
pBarScheme->Color1 = TLO_ATO3;           //RGB565CONVERT(150, 157, 36);
pBarScheme->EmbossDkColor = BLUE;         //RGB565CONVERT(55,55,55);
pBarScheme->EmbossLtColor = BLUE;         //RGB565CONVERT(0xAC, 0xB5, 0xB8);
pBarScheme->ColorDisabled = BLACK;
pBarScheme->TextColor1 = WHITE; //BLUE;    //RGB565CONVERT(255, 255, 255);
pBarScheme->TextColor0 = WHITE; //BLUE;    //RGB565CONVERT(255, 255, 255);
pBarScheme->TextColorDisabled = BLACK;

//*****
*****

SYSTEMConfigPerformance(GetSystemClock());
mOSCSetPBDIV(OSC_PB_DIV_1);
DDPCONbits.JTAGEN = 0; //pozwała na używanie bitów RB10 do RB13,

GOL_MSG msg; // definicja - GOL message structure to interact with GOL
GOLInit();    // initialize graphics library

spi_ch2_busy=0; // kanału 2 SPI, nie jest zajęty, przez AT45
Przerwania_Init ();
AT45_Init();
LCD_LED_INIT;
LCD_LED_ON;
UstawJasnosc(JASNOSC_LED_MIN);           //ekran ciemny
TouchInit();
Init_Wejsc_Wyjsc();
Init_ADS1115();
SetColor(BLUE);
ClearDevice();
OdczytKonfiguracji(); //m.in. odczytuje i podstawiam zmienną "jasnosc_led"
//===== przedłużona inicjalizacja MP3 =====
//po power on VS1011 trzymane w resecie przez rezystor ściąający-100k, noga procesora XRES jako
trzystanowa.
//cd.. inicjalizacji podczas wyświetlania ekranu startowego
//=====
unsigned int cykl=0,stop=0;
unsigned int x=50, y=50;
//Zero_Milisek();
zmien_strone_graf=0;
// EnableWDT();           // watchdog - start
testowy_display=0;
while(1){

//=====
=====
//
//      ClearWDT();
//      OdczytStanuWejsc();           // istotny dla trybu serwisowego
if(GOLDraw()){ // Draw GOL objects, ==1 gdy rysowanie obiektów zakończone
//=== obsługa zmiany stron graficznych ===
if(zmien_strone_graf){
    zmien_strone_graf=0;
    if(strona_graf==1){
        strona_graf=2;
        PokazOkno2();
    } else {
        strona_graf=1;
        PokazOkno1();
    }
}
}

```



```

//ustaw podświetlenie zapamiętane w konfiguracji, po maksymalnie jasnej stronie
startowej,

//istotne w pierwszym wywołaniu, ale może zostać
UstawJasnosc(jasnosc_led);

    }
    TouchGetMsg(&msg); // odczyt i obróbka touch screen
    GOLMsg(&msg); // przetwarzanie zdarzeń z touch-ekranu
    //GetZdarzenieInput(); // odczyt wejść i przypisanie im zdarzeń
    //ProcesZdarzenieInput(); // przetwarzanie zdarzeń z wejść
    DisplayTupDot();
    if(testowy_display){//tryb serwisowy lub praca z podglądem
        //DispPresuryName();
        //DisplayPresury();
        DisplayStanOut();
        DisplayStanInt();
    }
}
if(tryb_serwisowy) SterowanieSerwisowe(); //testy serwisowe
else SterowanieWy();
main_cykl++;
}
}
WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER* pObj, GOL_MSG* pMsg){
    // beep na każdy przycisk
    //if(objMsg == BTN_MSG_PRESSED){ Beep(); }
    // process messages
    switch(ekran){
        case DISPLAY_STARTOWY: return MsgEkranStartowy(objMsg, pObj);
        case DISPLAY_SERWIS: return MsgEkranZero(objMsg, pObj);
        case DISPLAY_TESTY: return MsgEkranTesty(objMsg, pObj);
        case DISPLAY_MAIN: return MsgEkranMain(objMsg, pObj);
        case DISPLAY_ZOMBIEK: return MsgEkranuZombiek(objMsg, pObj);
        case DISPLAY_KANAL: return MsgEkranuKanal(objMsg, pObj);
        case DISPLAY_BUTLA: return MsgEkranuButla(objMsg, pObj);
        case DISPLAY_SZKLANKA: return MsgEkranuSzklanka(objMsg, pObj);
        case DISPLAY_CONFIG: return MsgEkranConfig(objMsg, pObj);
        case DISPLAY_OKIENKO: return MsgEkranuOkienko(objMsg, pObj);
        //case DISPLAY_MP3: return MsgEkranTestMP3(objMsg, pObj);

        //-----
        //-----
        //-----
        //case DISPLAY_PLUKANIE_O2: ; //return MsgEkranPlukanieO2(objMsg,
pObj);
        //case DISPLAY_SPUST_CISN: ; //return MsgEkranSpustCis(objMsg, pObj);
        //case DISPLAY_SPUST_CISN_2: ; //return MsgEkranSpustCis(objMsg,
pObj);
        //case DISPLAY_ODWILGACANIE: return MsgEkranTechniczny(objMsg, pObj);
        //MsgEkranOdwilzanie(objMsg, pObj);
        //-----
        //-----
        default: return 1;
    }
}
WORD GOLDDrawCallback(){

    //przerysowania główne
    switch(ekran){

```

```

        case CREATE_STARTOWY:          CreateEkranStartowy();   ekran = DISPLAY_STARTOWY;   break;

        case CREATE_SERWIS:            CreateEkranASerwis();     ekran = DISPLAY_SERWIS;     break;

        case KALIBRACJA_TOUCH:         KalibracjaToucha();      ekran = CREATE_SERWIS;     break;
        //case CREATE_MP3:              CreateEkranTestMP3();     ekran = DISPLAY_MP3;

break;
        case CREATE_TESTY:
        CreateEkranATesty();
        ekran = DISPLAY_TESTY;
        tryb_serwisowy=1;
        break;

case CREATE_MAIN:                    CreateEkranMain();          ekran = DISPLAY_MAIN;   break;
case CREATE_ZOMBEC:                  CreateEkranZombek();        ekran = DISPLAY_ZOMBEC;   break;
case CREATE_KANAL:                   CreateEkranKanal();         ekran = DISPLAY_KANAL;
        break;
case CREATE_BUTLA:                   CreateEkranButla();         ekran = DISPLAY_BUTLA;
        break;
case CREATE_SZKLANKA:                CreateEkranSzklanka();      ekran = DISPLAY_SZKLANKA; break;
        //-----
case CREATE_OKIENKO:                 CreateEkranOkienko();       ekran = DISPLAY_OKIENKO; break;

case CREATE_CONFIG:                  CreateEkranConfig();        ekran = DISPLAY_CONFIG;   break;
        //-----

        //----- serwisowe
        //case CREATE_PLUKANIE_O2:      CreateEkranPlukanieO2(); ekran = DISPLAY_PLUKANIE_O2;   break;
        //-----
        //case CREATE_SPUST_CISN:       CreateEkranSpustCis();      ekran =
DISPLAY_SPUST_CISN;          break;
        //case CREATE_SPUST_CISN_2:     CreateEkranSpustCis_2();   ekran = DISPLAY_SPUST_CISN_2;   break;
        //case CREATE_ODWILGACANIE:     CreateEkranOdwilzanie();   ekran = DISPLAY_ODWILGACANIE;   break;
        //-----
        case CREATE_INFO:               CreateEkranInfo();          ekran =
DISPLAY_INFO;               break;

        //----- zmiany - odświeżenie informacji na ekranach
        case DISPLAY_STARTOWY:          CzekajEkranStartowy();      break;
        case DISPLAY_SERWIS:            DisplayTup();                break;
        //wyświetla napisik "TUP"
        case DISPLAY_MAIN:              ZmianyEkranuMain();          break;
        case DISPLAY_ZOMBEC:            ZmianyEkranuZombek();        break;
        case DISPLAY_KANAL:             ZmianyEkranuKanal();         break;
        case DISPLAY_BUTLA:             ZmianyEkranuButla();         break;
        case DISPLAY_SZKLANKA:          ZmianyEkranuSzklanka();      break;
        case DISPLAY_OKIENKO:          ZmianyEkranuOkienko();        break;
        case DISPLAY_INFO:              ZmianyEkranuInfo();          break;

        //case DISPL_KALI:               /*Touch_kalibracja()*/     ekran = CREATE_SERWIS;   break;
        case DISPL_LOAD_BMP:            ImportDaty(1);              ekran = CREATE_SERWIS;   break;
        case DISPL_LOAD_MP3:            ImportDaty(2);              ekran = CREATE_SERWIS;   break;

case DISPLAY_TESTY:
        DisplayPresury();          //wyświetla wartości ciśnień
        DisplayInputy();          //wyświetla stany 5-ciu wejść
        DisplayTup();              //wyświetla napisik "TUP"
        break;

```

```

//case DISPLAY_PLUKANIE_O2:                ZmianyEkranuPlukanieO2();                break;

//case DISPLAY_SPUST_CISN:                ZmianyEkranuSpustCis();                break;
//case DISPLAY_SPUST_CISN_2:                ZmianyEkranuSpustCis_2();                break;

//case DISPLAY_ODWILGACANIE:                ZmianyEkranuOdwilganie();                break;
}
//===== przerysowywanie według ustawionych flag =====
if(animacje_aktywne){
    //flaga fazy animacji zegar z funkcje_hardware ustawiana co 10 msek
    if(stan_O3==STAN_ON){
        //do wykorzystania na animacje
    }
    if(stan_butli== BUTLA_O3_GRANICA){
        if(faza_animacji==15 ){PutImage(286,4, (void*)&ico_butelka_szara8,IMAGE_NORMAL); }
        if(faza_animacji==40 ){PutImage(286,4, (void*)&ico_butelka8,IMAGE_NORMAL); }
    }
    faza_animacji++;
    if(faza_animacji==50 ){
        faza_animacji=0; //bez odświeżania ikon w pętli, a tylko po Create...
    }
    // wyświetlanie czasu i stanu butli również zależne od aktywności animacji
    if(FlagaZmianyCzasuButli()){
        KasujFlageZmianyCzasuButli();
        rysuj_czas_butli=1;                // flaga rysowania ustawiana dodatkowo przy
przerysowywaniu ikon
        switch(stan_butli){
            //case BUTLA_O3_BRAK: break;
            case BUTLA_O3_GRANICA:
                if(TmrButli()>=CZAS_BUTLA_O3BRAK){stan_butli = BUTLA_O3_BRAK;
rysuj_butelka=1; }
                break;
            case BUTLA_O3_OK:
                if(TmrButli()>=CZAS_BUTLA_O3DOBRE){stan_butli = BUTLA_O3_GRANICA;}
                break;
        }
    }
}
if(rysuj_ikony){
    rysuj_ikony=0;
    rysuj_piorun=1;
    rysuj_butelka=1;
    rysuj_czas_butli=1;
}
if(rysuj_piorun){
    rysuj_piorun=0;
    if(stan_O3==STAN_ON) {PutImage(160,3, (void*)&ico_piorun_luz8,IMAGE_NORMAL); }
    if(stan_O3==STAN_OFF){PutImage(160,3, (void*)&ico_piorun_tlo8,IMAGE_NORMAL); }
}
if(rysuj_butelka){
    rysuj_butelka=0;
    if(stan_butli==BUTLA_O3_BRAK){PutImage(286,4, (void*)&ico_butelka_szara8,IMAGE_NORMAL); }
    if(stan_butli==BUTLA_O3_OK) {PutImage(286,4, (void*)&ico_butelka8,IMAGE_NORMAL); }
}
if(rysuj_czas_butli){                // rysuj_czas_butli{
    WORD czas;

```

```

char min,sek,dm,jm,ds,js;
static char sdm=0,sjm=0,sds=0,sjs=0;
rysuj_czas_butli=0;
czas=CZAS_BUTLA_O3BRAK-TmrButli();
if(stan_butli == BUTLA_O3_BRAK){czas=0;};
min=czas/60;    // minuty
sek=czas%60;    // sekundy
dm=min/10;
jm=min%10;
ds=sek/10;
js=sek%10;
DisplayCzasButli(sdm,sjm,sds,sjs, TLO_GREEN);
DisplayCzasButli(dm,jm,ds,js, SZARA_CYFRA);
sdm=dm;        sjm=jm; sds=ds; sjs=js;
}

if(rys_jasnosc_led){
    //DrawSuwak(YELLOW,172,80,185,200,40,jasnosc_led);
    //DispSuwakV(172,80,200,40,jasnosc_led);
    disp_jasnosc_led=jasnosc_led;
    rys_jasnosc_led=0;
}

if(rys_jezyk_flaga){
    switch(jezyk){
        case POLSKI:
            SetColor(BLUE);
            Bevel(85,73,132,98,5);
            Bevel(86,74,131,97,5);
            SetColor(GREGGREY);
            Bevel(165,73,212,98,5);
            Bevel(166,74,211,97,5);
            break;
        case ANGIELSKI:
            SetColor(GREGGREY);
            Bevel(85,73,132,98,5);
            Bevel(86,74,131,97,5);
            SetColor(BLUE);
            Bevel(165,73,212,98,5);
            Bevel(166,74,211,97,5);
            break;
    }
    rys_jezyk_flaga=0;
}

/*
if(rys_volume_dzwiek){
    DrawSuwak(BLUE,172,80,185,200,40,volume_dzwiek);
    disp_volume_dzwiek=volume_dzwiek;
    rys_volume_dzwiek=0;
}
*/

if(rysuj_czas_sekundy){
    DisplayCzasSekundy(disp_czas_sekundy, GREGWHITE);
    DisplayCzasSekundy(czas_sekundy, SZARA_CYFRA);
    disp_czas_sekundy=czas_sekundy;
    rysuj_czas_sekundy=0;
}

```

```

        return 1;
    }

WORD ExternalMemoryCallback(EXTDATA* memory, LONG offset, WORD nCount, void* buffer){
    return nCount;
}

//==== funkcje pomocnicze, wewnętrzne modułu
=====

//=== TextSel - zwraca wskaźnik na tekst w odpowiedniej wersji językowej
ptext TextSel(typ_Język lang, typ_Napis text_num){
    if(lang==POLSKI){return TextPol[text_num];}
    if(lang==ANGIELSKI){return TextAng[text_num];}
}

//=== OutTextXJustedY == wyświetla tekst wyjustowany względem środka ekranu, z przesunięciem deltax
//justowanie w funkcji OutTextXJustedY(..)
//deklaracje w pliku *ato3.h - dzielniki szerokości wyświetlanego stringu
//define RIGHT_JUSTED 1 //przesunięcie o całą szerokość, tekst dosunięty prawą stroną do środka
//define CENTRE_JUSTED 2 //przesunięcie o 1/2 szerokości, tekst symetryczny względem środka ekranu
void OutTextXJustedY(SHORT deltax,SHORT y, char just, XCHAR * my_string, void * font){
    SHORT width;
    width = GetTextWidth(my_string, font);
    if(just==0) just =1;
    OutTextXY(160+deltax-width/just,y,my_string);
}

void DisplayTup(){
    SetFont((void*)&GOLMediumFont);
    if(TupTouch()){
        SetColor(WHITE);
        OutTextXY(290,195,"TUP");
    }
    else{
        SetColor(BLACK);
        OutTextXY(290,195,"TUP");
    }
}

void DisplayTupDot(){
    //SetFont((void*)&GOLMediumFont);
    if(TupTouch()){SetColor(BLACK);}
    else {SetColor(WHITE);}
    PutPixel(54,26);
    PutPixel(56,26);
    //Line(54,27,55,27);
}

void DispPresuryName(){
    SetColor(WHITE);
    SetFont((void*)&GOLMediumFont);
    OutTextXY(100,5,"ABS:");
    OutTextXY(210,5,"DIFF:");
}

void DisplayPresury(){
    SetFont((void*)&GOLMediumFont);
    if( Flaga_Abs()){
        Kasuj_Flage_Abs();

        SetColor(TLO_ATO3);
        LiczbaToText(pomiar_abs,6,&text_bufor[0]);
        OutTextXY(140,5,&text_bufor[0]);
    }
}

```

```

        pomiar_abs=Cisnienie_Abs();

SetColor(WHITE);
    LiczbaToText(pomiar_abs,6,&text_bufor[0]);
    OutTextXY(140,5,&text_bufor[0]);
}
if( Flaga_Diff()){
    Kasuj_Flage_Diff();

SetColor(TLO_ATO3);
    LiczbaToText(pomiar_diff,6,&text_bufor[0]);
    OutTextXY(250,5,&text_bufor[0]);

    pomiar_diff=Cisnienie_Diff();

SetColor(WHITE);
    LiczbaToText(pomiar_diff,6,&text_bufor[0]);
    OutTextXY(250,5,&text_bufor[0]);
}
}
void DiodyRGB_On(void){
    switch (stezenie_robocze){
        case 3: stan_blue =STAN_ON; break;        //stan_blue =STAN_ON;stan_blue =STAN_ON;
        case 2: stan_green =STAN_ON; stan_blue=STAN_ON; break;
        case 1: stan_green =STAN_ON; break;
    }
}
void DiodyRGB_Off(void){
    stan_red =STAN_OFF;
    stan_green=STAN_OFF;
    stan_blue =STAN_OFF;
}
void DisplayStanInt(){
    if(nogal_yellow){SetColor(YELLOW); Bar(50,2,60,12);} else {SetColor(SZARA_CYFRA); Bar(49,1,68,13);}
    if(nogal_blue) {SetColor(LIGHTBLUE); Bar(80,2,90,12);} else {SetColor(SZARA_CYFRA); Bar(72,1,91,13);}
}
void DisplayStanOut(){
    SetFont((void*)&GOLMediumFont);
    if(stan_V1){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(110,2,"V1");
    if(stan_V2){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(140,2,"V2");
    if(stan_V3){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(170,2,"V3");
    if(stan_V4){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(200,2,"V4");
    if(stan_V5){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(230,2,"V5");
    if(stan_O3){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(120,20,"O3");
    if(stan_PK1){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(150,20,"PK1");
    if(stan_PK2){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(180,20,"PK2");
    if(stan_pompa1){SetColor(WHITE);} else {SetColor(BLACK);}
    OutTextXY(210,20,"Pmp");
    //if(stan_pompa2){SetColor(WHITE);} else {SetColor(BLACK);}
    //OutTextXY(240,20,"Pp2");
}
}

```



```

void DisplayZmianySTE(int poz){
    static int ppoz=0;
    SetFont((void*)&GOLMediumFont);
    SetColor(BLUE);
    LiczbaToText(ppoz,4,&text_bufor[0]);
    OutTextXY(275,95,&text_bufor[0]);
    SetColor(WHITE);
    LiczbaToText(poz,4,&text_bufor[0]);
    OutTextXY(275,95,&text_bufor[0]);
    ppoz=poz;
}

void DisplayInputy(){
    if(nogal_yellow){SetColor(RED); Bar(190,220,200,230);}      else {SetColor(WHITE); Bar(188,218,202,232);}
    if(nogal_blue)    {SetColor(RED); Bar(240,220,250,230);}    else {SetColor(WHITE); Bar(238,218,252,232);}
}

void DisplayOutyName(){
    SetColor(WHITE);
    SetFont((void*)&GOLMediumFont);
    OutTextXY(10,75,"- -,P4");
    OutTextXY(80,75,"- -,P3");
    OutTextXY(150,75,"- -,P1");
    OutTextXY(220,75,"- -,P2");
    OutTextXY(10,130,"Lp1,P5");
    OutTextXY(80,130,"Lp2,P6");
    OutTextXY(150,130,"Fan,P7");
    OutTextXY(220,130,"- -,P8");
    OutTextXY(10,185,"rejstry latch 1, 2, 3");
    OutTextXY(220,185,"Out_CS");
}

char AnyKey(void){

static char any = 0;
char stan;

    stan = TupTouch();
    if(any != stan){
        if(any){any=0; return 1;}
        any = stan;
    }

    return 0;
}

/*
void PlayMP3(EXTDATA* sound){
    if(VS1011_Softreset()){          //softreet zwraca zero gdy nie halo
        UstawVolume(volume_dzwiek);
        VS1011_Config();
        if(stan_mp3) StartPlayMP3(sound);
    }
}

void Klik(){
    //StartPlayMP3(&klik);
    //PlayMP3(&klik);
    //PlayMP3(&klementyna);
}

*/
//===== ELEMENTY CREATE-EKRANOW =====

/*
    opis sygnałów magistrali wyjściowej

```

sygnal	LE_1	LE_2,	LE_3
xx_WY_1	--,	P8,	V6
xx_WY_2	--,	P7,	V5
xx_WY_3	--,	P6,	V4
xx_WY_4	--,	P5,	V3
xx_WY_5	----,	P2,	O3
xx_WY_6	----,	P1,	PMP
xx_WY_7	--,	P3,	V2
xx_WY_8	--,	P4,	V1

```

*/
//===== BUTONY - serwisowe - zmiennalne =====
void Button_V1_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_V1, 10, 40, 50, 75, 5, BTN_DRAW, NULL, V1Str, pScheme);
}
void Button_V2_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_V2, 70, 40, 110, 75, 5, BTN_DRAW, NULL, V2Str, pScheme);
}
void Button_V3_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_V3, 130, 40, 170, 75, 5, BTN_DRAW, NULL, V3Str, pScheme);
}
void Button_V4_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_V4, 190, 40, 230, 75, 5, BTN_DRAW, NULL, V4Str, pScheme);
}
void Button_V5_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_V5, 250, 40, 290, 75, 5, BTN_DRAW, NULL, V5Str, pScheme);
}
//---
void Button_Pompa1_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_POMPA1, 10, 100, 50, 135, 5, BTN_DRAW, NULL, "Pm1", pScheme);
}
void Button_Fan_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_FAN, 70, 100, 110, 135, 5, BTN_DRAW, NULL, "Fan", pScheme);
}
void Button_O3_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_O3, 130, 100, 170, 135, 5, BTN_DRAW, NULL, O3Str, pScheme);
}
void Button_PK1_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_PK1, 190, 100, 230, 135, 5, BTN_DRAW, NULL, "PK1", pScheme);
}
void Button_PK2_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_PK2, 250, 100, 290, 135, 5, BTN_DRAW, NULL, "PK2", pScheme);
}
//---
void Button_R_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_RED, 10, 160, 50, 195, 0, BTN_DRAW, NULL, "R", pScheme);
}
void Button_G_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_GREEN, 70, 160, 110, 195, 0, BTN_DRAW, NULL, "G", pScheme);
}
void Button_B_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_BLUE, 130, 160, 170, 195, 0, BTN_DRAW, NULL, "B", pScheme);
}
/*
void Button_Pompa2_create (GOL_SCHEME *pScheme) {
    BtnCreate( ID_BUTTON_POMPA2, 70, 100, 110, 135, 5, BTN_DRAW, NULL, "Pm2", pScheme);
}
void Button_T_plus_create (uint16_t status) {
    BtnCreate( ID_BUTTON_TPLUS, 280, 40, 310, 80, 0, status, NULL, PlusStr, BtnScheme);
}

```



```

}
void Button_T_minus_create (uint16_t status) {
    BtnCreate( ID_BUTTON_TMINUS, 280, 130, 310, 170, 0, status, NULL, MinusStr, BtnScheme);
}
*/
void Button_Back_create () {
    BtnCreate( ID_BUTTON_BACK, 5, 5, 65, 30, 0, BTN_DRAW, NULL, BackStr, BtnScheme);
}
void DisplayMinutyOzon(char czas, int kolor){
    SetColor(kolor);
    //SetFont((void*)&Tahoma_cyfry_80);
    SetFont((void*)&Engravers_16);
    LiczbaToText(czas,2,&text_bufor[0]);
    OutTextXJustedY(25,127,CENTRE_JUSTED,&text_bufor[0],(void*)&Tahoma_cyfry_80);
}
void DisplayCzasSekundy(char czas, int kolor){
    SetColor(kolor);
    SetFont((void*)&Engravers_16);
    LiczbaToText(czas,2,&text_bufor[0]);
    OutTextXJustedY(20,127,CENTRE_JUSTED,&text_bufor[0],(void*)&Engravers_16);
}
void DisplayCzas_9_59(char min,char dsek,char jsek, int kolor){
    //WORD min, sek, dsek, jsek;
    SetColor(kolor);
    SetFont((void*)&Engravers_16);
    Bar(165,147,167,149);
    Bar(165,157,167,159);
    LiczbaToText(min,1,&text_bufor[0]);
    OutTextXY(134,127,&text_bufor[0]);
    LiczbaToText(dsek,1,&text_bufor[0]);
    OutTextXY(169,127,&text_bufor[0]);
    LiczbaToText(jsek,1,&text_bufor[0]);
    OutTextXY(198,127,&text_bufor[0]);
}
void DisplayCzasButli(char dmin, char jmin, char dsek, char jsek, int kolor){
    //WORD min, sek, dsek, jsek;
    SetColor(kolor);
    //SetFont((void*)&GOLMediumFont); //Engravers_8);
    SetFont((void*)&Arial_bold_6);
    PutPixel(250,16);
    PutPixel(250,20);
    LiczbaToText(dmin,1,&text_bufor[0]);
    OutTextXY(222,7,&text_bufor[0]);
    LiczbaToText(jmin,1,&text_bufor[0]);
    OutTextXY(237,7,&text_bufor[0]);
    LiczbaToText(dsek,1,&text_bufor[0]);
    OutTextXY(255,7,&text_bufor[0]);
    LiczbaToText(jsek,1,&text_bufor[0]);
    OutTextXY(270,7,&text_bufor[0]);
}
void DisplayMinSek(SHORT x, SHORT y){
    //SetFont((void*)&Arial_6);
    //SetFont((void*)&Tahoma_cyfry_80);
    SetFont((void*)&Engravers_16);
    if(czas_dm){
        LiczbaToText(czas_dm,1,&text_bufor[0]);
        OutTextXY(x-58,y,&text_bufor[0]);
    }
    LiczbaToText(czas_jm,1,&text_bufor[0]);

```

```

    OutTextXY(x-31,y,&text_bufor[0]);
    //bary za dwukropek
    Bar(x,y+20,x+2,y+22);
    Bar(x,y+30,x+2,y+32);

    LiczbaToText(czas_ds,1,&text_bufor[0]);
    OutTextXY(x+3,y,&text_bufor[0]);
    LiczbaToText(czas_js,1,&text_bufor[0]);
    OutTextXY(x+33,y,&text_bufor[0]);
}

//===== pomocnicze =====
/*
WORD StezeniePrzekaznikowe(WORD stez){
    switch(stez){

        case 5:  return OZON_LEVEL_5;
        case 10: return OZON_LEVEL_10;
        case 15: return OZON_LEVEL_15;
        case 20: return OZON_LEVEL_20;
        case 30: return OZON_LEVEL_30;
        case 40: return OZON_LEVEL_40;
        case 50: return OZON_LEVEL_50;
        case 60: return OZON_LEVEL_60;
        case 70: return OZON_LEVEL_70;
        default: return OZON_LEVEL_5;

    }
}
*/
void UstawStezenieRobocze(WORD stez){
    //aktualizuje zmienne stanu przekazników stężenia
    switch (stezenie_robocze){
        case 1:  stan_PK1=STAN_OFF;    ;stan_PK2=STAN_OFF;        break;
        case 2:  stan_PK1=STAN_OFF;    ;stan_PK2=STAN_ON;         break;
        case 3:  stan_PK1=STAN_ON;     ;stan_PK2=STAN_ON;         break;

    }
}

void PrzeliczenieCzasu(WORD czas){
    czas_jh=0;
    if(czas>=3600){
        czas_jh=czas/3600;
        czas=czas-3600*czas_jh;
    }
    czas_dm=0;
    if(czas>=600){
        czas_dm=czas/600;
        czas=czas-600*czas_dm;
    }
    czas_jm=0;
    if(czas>=60){
        czas_jm=czas/60;
        czas=czas-60*czas_jm;
    }
    czas_ds=0;
    if(czas>=10){
        czas_ds=czas/10;
        czas=czas-10*czas_ds;
    }
}

```

```

        czas_js=czas;
    }
    //=====
void PokazCzas(SHORT x, SHORT y, char h, char min){
    //SetFont((void*)&GOLFontDefault);
    SetFont((void*)&Tahoma_cyfry_80);
    char z=0;
    if(h){
        LiczbaToText(czas_jh,1,&text_bufor[0]);
        OutTextXY(x,y,&text_bufor[0]);
        OutTextXY(x+13,y,":");
    }
    else z=20;
    if(min){
        LiczbaToText(czas_dm,1,&text_bufor[0]);
        OutTextXY(x+20-z,y,&text_bufor[0]);
        LiczbaToText(czas_jm,1,&text_bufor[0]);
        OutTextXY(x+33-z,y,&text_bufor[0]);
        OutTextXY(x+46-z,y,":");
    }
    else z=53;
    LiczbaToText(czas_ds,1,&text_bufor[0]);
    OutTextXY(x+53-z,y,&text_bufor[0]);
    LiczbaToText(czas_js,1,&text_bufor[0]);
    OutTextXY(x+66-z,y,&text_bufor[0]);
}

void PokazHourMinSek(SHORT x, SHORT y, char h, char min, char sek){
    SetFont((void*)&GOLFontDefault);
    char z=0;
    if(h){
        LiczbaToText(czas_jh,1,&text_bufor[0]);
        OutTextXY(x,y,&text_bufor[0]);
        OutTextXY(x+13,y,":");
    }
    else z=20;
    if(min){
        LiczbaToText(czas_dm,1,&text_bufor[0]);
        OutTextXY(x+20-z,y,&text_bufor[0]);
        LiczbaToText(czas_jm,1,&text_bufor[0]);
        OutTextXY(x+33-z,y,&text_bufor[0]);
        OutTextXY(x+46-z,y,":");
    }
    else z=53;
    if(sek){
        LiczbaToText(czas_ds,1,&text_bufor[0]);
        OutTextXY(x+53-z,y,&text_bufor[0]);
        LiczbaToText(czas_js,1,&text_bufor[0]);
        OutTextXY(x+66-z,y,&text_bufor[0]);
    }
}

//=====
void CreateEkranStartowy(){
    GOLFree();
    PutImageExt(&ekr_start, 0, 0);    //
    //po wyświetleniu bitmapy włącz podświetlenie LED na full
    UstawJasnosć(JASNOSC_LED_MAX);
    //lub narastanie jasności
    SetFont((void*)&Verdana_standard_12);
    SetColor(BLUE);
}

```

```

        OutTextXJustedY(0, 4, CENTRE_JUSTED, TextSel(jezyk,Aparatdo), (void*)&Verdana_standard_12);
        // button ID, dimension ,radius, status, bitmap, text, color scheme//
        //przycisk wywołania trybu serwisowego
        BtnCreate(ID_BUTTON1, 130,5,190,25, 0,          BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
        stan_mp3=1;
        ZeroCzasEtapu();
    }
    void CzekaJEkranStartowy(){
    //      WORD xxx;
        //xxx=CzasEtapuCs()/6; //narastanie jasności z czasem, o jeden poziom co 6/100 sekundy
        //if(xxx>JASNOSC_LED_MAX) xxx=JASNOSC_LED_MAX;
        //UstawJasnosc(xxx); //zastąpić narastaniem w przrwaniu i tablicą jasności
        //opóźniona inicjalizacja MP3, po kilkunastu milisekundach w resecie
        /*
        switch(stan_mp3){
            case 1: AUDIO_INIT; VS1011_Init(); stan_mp3=2; break;
            case 2: if(VS1011_DREQ==1) {stan_mp3=3;} break;
            case 3: VS1011_SetVolume (254); stan_mp3=4; break; //maksymalnie wyciszone, 255 to
analog powerdown
            case 4: VS1011_Config(); stan_mp3=5; break;
            case 5: StartPlayMP3(&klementyna); stan_mp3=6; ZeroCzasEtapu(); break;
            case 6: UstawVolume(volume_dzwiek); stan_mp3=7; break;
            //case 6: VS1011_SetVolume (23); stan_mp3=7; break; // a tak działa
        }
        */
        if(GetCzasZabiegu() > CZAS_STARTU){
            ekran=CREATE_MAIN;
            if(stan_mp3!=7) stan_mp3=0; //wyłącz odtwarzanie dźwięków, VS1011 nie działa
        }
    }
    //-----
    WORD MsgEkranStartowy(WORD objMsg, OBJ_HEADER* pObj){
        OBJ_HEADER* pOtherRbtn;

        switch(GetObjID(pObj)){
            case ID_BUTTON1:
                if(objMsg == BTN_MSG_RELEASED){          ekran=CREATE_SERWIS;          }
                return 1;
            default:
                return 1;
        }
    }
    //-----
    void CreateEkranASerwis(){
        GOLFree();
        SetColor(BLACK);
        ClearDevice();
        SetColor(BLUE);
        Bar(0,40,319,170);
        tryb_serwisowy=0;
        //DispPresuryName(); // wyświetla jednorazowo nagłówki
        ciśnien

        SetColor(BLUE);
        Bar(0,40,319,170);
        // "Testy",
        BtnCreate(ID_BUTTON1,          30, 50,120,100,          10,          BTN_DRAW, NULL, "testy", altScheme);

        //BtnCreate(ID_BUTTON2,200,110,290,160, 10,          BTN_DRAW,          NULL, "sound", altScheme);
    
```

```

        // "praca normalna z podglądem we/wy",    // text
BtnCreate(ID_BUTTON3,      200,170,290,230, 10,    BTN_DRAW, NULL, "nano", altScheme);
    // "KalTouch",    // text
BtnCreate(ID_BUTTON4,      30,110,120,160,      10,    BTN_DRAW, NULL, "touch", altScheme);
    // "BMP",    // text
BtnCreate(ID_BUTTON5,      200, 50,290,100, 10,    BTN_DRAW,    NULL, "import", altScheme);
    // "MP3",    // text
//BtnCreate(ID_BUTTON6,      200, 5,290, 40,  10,    BTN_DRAW,    NULL, "im-mp3", altScheme);

char id=0;
SetColor(WHITE);
    SetFont( (void*)&Verdana_standard_10 );
    OutTextXY(10,3,S_Number);
    SetFontOrientation(1);    // do góry
    OutTextXY(8,130,"Testy"); //
    SetFontOrientation(2);    // na dół
    OutTextXY(315,80,"Testeczki");    //odpowiednio zmodyfikowana funkcja OutChar w Primitive.c
    SetFontOrientation(0);
    OutTextXY(10,170,"Producent");    //
    id=Memo_Id_Read(1);
    LiczbaToText(id,3,&text_bufor[0]);
    OutTextXY(100,170,&text_bufor[0]);
    OutTextXY(10,190,"DataFlash");    //
    id=Memo_Id_Read(2);
    LiczbaToText(id,3,&text_bufor[0]);
    OutTextXY(100,190,&text_bufor[0]);
    OutTextXY(10,210,"Page");    //bit 0=0-1056, =1-1024
    id=status_read();
    if((id&0x01)==1) OutTextXY(100,210,"1024");
    else OutTextXY(100,210,"1056");
}
//-----
WORD MsgEkranZero(WORD objMsg, OBJ_HEADER* pObj){
    OBJ_HEADER* pOtherRbtn;

    switch(GetObjID(pObj)){
        case ID_BUTTON1:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_TESTY; }
            return 1;
        case ID_BUTTON2:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_MP3; }
            return 1;
        case ID_BUTTON3:
            if(objMsg == BTN_MSG_PRESSED){} // "praca normalna z podglądem we/wy",
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_MAIN; testowy_display=1;}
            return 1;
        case ID_BUTTON4:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ ekran=KALIBRACJA_TOUCH;} //DISPL_KALI;}
            return 1;
        case ID_BUTTON5:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ ekran=DISPL_LOAD_BMP; }
            return 1;
        case ID_BUTTON6:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ ekran=DISPL_LOAD_MP3; }
    }
}

```

```

        return 1;

default:
    return 1;                                // process by default
}
}
//=====================================================
void AktywacjaMP3(void){
    unsigned int w=0;
    unsigned int m=0;
    unsigned int ugi;
    AUDIO_INIT;
    w = VS1011_Init();
    if(w==0){
        ZeroUsek();    while(GetUsek()<100000){};    //czekam w resecie 50msek
        w = VS1011_Init(); m=1;
    }
    if(w==0){
        ZeroUsek();    while(GetUsek()<100000){};    //czekam w resecie 50msek
        w = VS1011_Init(); m=2;
    }
    SetColor(BLACK);
    SetFont( (void*)&Verdana_standard_10 );
    OutTextXY(100, 25, "reset hard");
    LiczbaToText(w,9,&text_bufor[0]);
    OutTextXY(200, 25, &text_bufor[0]);
    LiczbaToText(m,2,&text_bufor[0]);
    OutTextXY(60, 25, &text_bufor[0]);
    /*
    w=VS1011_Softreset();

    */
    w=VS1011_SetVolume (254); //zupełna cisza, 255 = analog powerdown mode
    OutTextXY(180, 40, "volume");
    LiczbaToText(w,6,&text_bufor[0]);
    OutTextXY(260, 40, &text_bufor[0]);
    m=VS1011_Config();
    OutTextXY(160, 170, "konfig");
    LiczbaToText(m,3,&text_bufor[0]);
    OutTextXY(260, 170, &text_bufor[0]);

    w=VS1011_GetVolume ();
    OutTextXY(160, 190, "volume set");
    LiczbaToText(w,3,&text_bufor[0]);
    OutTextXY(260, 190, &text_bufor[0]);
}

void StartMP3(void){
    unsigned int w=7;
    unsigned int m=8;
    unsigned int ugi;
    AUDIO_INIT;
    w = VS1011_Init();
    OutTextXY(100, 60, "reset hard");
    LiczbaToText(w,6,&text_bufor[0]);
    OutTextXY(200, 60, &text_bufor[0]);

    w=VS1011_Softreset();
    OutTextXY(100, 80, "softreset");
    LiczbaToText(w,6,&text_bufor[0]);
    OutTextXY(200, 80, &text_bufor[0]);
}

```

```

VS1011_SetVolume (65);
delay_ms();

m=VS1011_Config();
OutTextXY(100, 100, "konfig");
LiczbaToText(m,3,&text_bufor[0]);
OutTextXY(200, 100, &text_bufor[0]);

w=VS1011_GetVolume ();
OutTextXY(100, 120, "volume set");
LiczbaToText(w,3,&text_bufor[0]);
OutTextXY(200, 120, &text_bufor[0]);
}

void DoTestCoj(){
    unsigned int w=0;
    SetColor(TLO_ATO3);
    Bar(100,55,290,160);           //pasek
    SetColor(WHITE);
    SetFont( (void*)&Verdana_standard_10 );
    AUDIO_INIT;
    InitMP3();
    delay_ms();
    VS1011_SetVolume (35);
    delay_ms();
    AUDIO_ON;

    ZeroLicznik();
    StartPlayMP3(&klik);
    OutTextXY(100, 120, "czas intT5 =");
    while(GramTeraz());
    w=Licznik();
    LiczbaToText(w,8,&text_bufor[0]);
    OutTextXY(200, 120, &text_bufor[0]);

    ZeroLicznik();
    StartPlayMP3(&klementyna);
    ZeroCzasEtapu();
    while(GramTeraz()){
        SetColor(WHITE);
        if(CzasEtapuCs()==5) Bar(160,60,170,80);
        if(CzasEtapuCs()==10) Bar(170,80,190,90);
        if(CzasEtapuCs()==15) Bar(160,90,170,110);
        if(CzasEtapuCs()==20) Bar(140,80,160,90);
        if(CzasEtapuCs()==20) OutTextXY(200, 75, "Akuku");
        SetColor(BLUE);
        if(CzasEtapuCs()==25) Bar(160,60,170,80);
        if(CzasEtapuCs()==30) Bar(170,80,190,90);
        if(CzasEtapuCs()==35) Bar(160,90,170,110);
        if(CzasEtapuCs()==40) Bar(140,80,160,90);
        if(CzasEtapuCs()==40){ OutTextXY(200, 75, "Akuku"); ZeroCzasEtapu();}
    }
    w=Licznik();
    SetColor(WHITE);
    OutTextXY(100, 140, "czas usek =");
    LiczbaToText(w,8,&text_bufor[0]);
    OutTextXY(200, 140, &text_bufor[0]);
}

```



```

        AUDIO_OFF;
    }

void DoTestMP3(){
    unsigned int w=7;
    unsigned int m=8;
    unsigned int ugi;
    static char num=1;

    SetColor(TLO_ATO3);
    Bar(100,55,290,160);           //pasek
    SetColor(WHITE);
    SetFont( (void*)&Verdana_standard_10 );
    ZeroUsek();
    AUDIO_INIT;
    InitMP3();
    delay_ms();
    VS1011_SetVolume (35);
    delay_ms();
    w=VS1011_GetVolume ();
    OutTextXY(100, 80, "volume read");
    LiczbaToText(w,3,&text_bufor[0]);
    OutTextXY(200, 80, &text_bufor[0]);
    AUDIO_ON;
    OutTextXY(100, 100, "muzic ");
    w=Graj_MP3(&kasia);
    LiczbaToText(w,6,&text_bufor[0]);
    OutTextXY(200, 100, &text_bufor[0]);
    AUDIO_OFF;
    num++;
}

void DoTestMP3_sin(){
    unsigned int w=7;
    unsigned int m=8;
    unsigned int ugi;
    SetColor(TLO_ATO3);
    Bar(100,55,290,160);           //pasek
    SetColor(WHITE);
    SetFont( (void*)&Verdana_standard_10 );
    AUDIO_INIT;
    w = VS1011_Init();
    OutTextXY(100, 60, "reset hard");
    LiczbaToText(w,6,&text_bufor[0]);
    OutTextXY(200, 60, &text_bufor[0]);

    m=VS1011_Config();
    m=VS1011_Testy();
    OutTextXY(100, 80, "test sinus");
    LiczbaToText(m,3,&text_bufor[0]);
    OutTextXY(200, 80, &text_bufor[0]);

    VS1011_SetVolume (0xFE);
    VS1011_OnSinus();
    AUDIO_ON;
    VS1011_SetVolume (30);
    OutTextXY(100, 120, "ON sinus");
}

```



```

        ZeroUsek();
        while(GetUsek()<75000){};          //300 msek
        w=VS1011_GetVolume ();
        VS1011_SetVolume (0xFE); //-2 ??
        AUDIO_OFF;
        VS1011_OffSinus();
        OutTextXY(100, 140, "OFF sinus");

        OutTextXY(100, 100, "volume read");
        LiczbaToText(w,3,&text_bufor[0]);
        OutTextXY(200, 100, &text_bufor[0]);

    }

    void TestSinus_ON(){

        unsigned int m=7;
        SetColor(TLO_ATO3);
        Bar(100,55,290,160);                //pasek
        SetColor(WHITE);
        SetFont( (void*)&Verdana_standard_10 );
        AUDIO_INIT;
        m = VS1011_Init();
        OutTextXY(100, 60, "reset hard");
        LiczbaToText(m,6,&text_bufor[0]);
        OutTextXY(200, 60, &text_bufor[0]);

        VS1011_SetVolume (50);
        delay_ms();

        m=VS1011_Config();
        m=VS1011_Testy();
        OutTextXY(100, 80, "test sinus");
        LiczbaToText(m,3,&text_bufor[0]);
        OutTextXY(200, 80, &text_bufor[0]);

        m=VS1011_GetVolume ();
        OutTextXY(100, 100, "volume read");
        LiczbaToText(m,3,&text_bufor[0]);
        OutTextXY(200, 100, &text_bufor[0]);

        VS1011_SetVolume (0xFE);
        VS1011_OnSinus();
        AUDIO_ON;
        VS1011_SetVolume (50);

    }

    void TestSinus_OFF(){

        VS1011_SetVolume (0xFE);
        AUDIO_OFF;
        VS1011_OffSinus();

    }

    void TestBmp(){
        offset_adresu=4094976;
        ZeroCzasEtapu();
        PutImageExt(&flagi_blue, 0, 0);
        while(CzasEtapu()<3){};
        PutImageExt(&flagi, 0, 0);
    }

```

```

        while(CzasEtapu()<6){};
        offset_adresu=0;
        ekran=CREATE_MP3;
    }

//=====
/*
void CreateEkranTestMP3(){
    GOLFree();
    SetColor(BLACK);
    ClearDevice();
    SetColor(TLO_ATO3);
    Bar(0,0,319,34);          //pasek góra tło
    Bar(0,235,319,239);      //pasek dół
    Bar(0,55,319,215);
    SetColor(WHITE);
    SetFont((void*)&Verdana_standard_10 );
    SetFontOrientation(1);
    OutTextXY(10,150,"Testeczki");    //
    SetFontOrientation(2);
    OutTextXY(310,65,"MP-trojeczki"); //odpowiednio zmodyfikowana funkcja OutChar w Primitive.c
    SetFontOrientation(0);
    BtnCreate(ID_BUTTON1, 30, 50, 90,100, 10, BTN_DRAW, NULL, "play", altScheme);
    BtnCreate(ID_BUTTON2, 30,120, 90,170, 10, BTN_DRAW, NULL, "pip", altScheme);
    BtnCreate(ID_BUTTON3, 30,190, 90,230, 10, BTN_DRAW, NULL, "Back", altScheme);
    BtnCreate(ID_BUTTON4, 120,170,180,210, 10, BTN_DRAW, NULL, "sinON", altScheme);
    BtnCreate(ID_BUTTON5, 200,170,260,210, 10, BTN_DRAW, NULL, "sinOF", altScheme);
    BtnCreate(ID_BUTTON6, 100, 5,200, 35, 10, BTN_DRAW, NULL, "test bmp", altScheme);
    BtnCreate(ID_BUTTON7, 220, 5,280, 35, 10, BTN_DRAW, NULL, "Coj", altScheme);
    StartMP3();
}

WORD MsgEkranTestMP3(WORD objMsg, OBJ_HEADER* pObjj){
    OBJ_HEADER* pOtherRbtn;

    switch(GetObjID(pObjj)){
        case ID_BUTTON1:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ DoTestMP3(); }
            return 1;
        case ID_BUTTON2:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ DoTestMP3_sin(); }
            return 1;
        case ID_BUTTON3:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_SERWIS; }
            return 1;
        case ID_BUTTON4:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ TestSinus_ON(); }
            return 1;
        case ID_BUTTON5:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ TestSinus_OFF(); }
            return 1;
        case ID_BUTTON6:
            if(objMsg == BTN_MSG_PRESSED){}
            if(objMsg == BTN_MSG_RELEASED){ TestBmp(); }
    }
}

```

```

        return 1;
    case ID_BUTTON7:
        if(objMsg == BTN_MSG_PRESSED){}
        if(objMsg == BTN_MSG_RELEASED){    DoTestCoj();    }
        return 1;

    default:
        return 1;                                // process by default
    }
}
*/
//=====
void CreateEkranATesty(){
    GOLFree();
    SetColor(BLACK);
    ClearDevice();
    SetColor(TLO_ATO3);
    Bar(0,0,319,34);        //pasek góra tło
    Bar(0,235,319,239);    //pasek dół
    SetColor(LIGHTBLUE);
    Bar(0,55,319,180);
    DispPresuryName();
    //DisplayOutyName();
    if(stan_V1){    Button_V1_create(OutOnScheme);    }else{    Button_V1_create(OutOffScheme); }
    if(stan_V2){    Button_V2_create(OutOnScheme);    }else{    Button_V2_create(OutOffScheme); }
    if(stan_V3){    Button_V3_create(OutOnScheme);    }else{    Button_V3_create(OutOffScheme); }
    if(stan_V4){    Button_V4_create(OutOnScheme);    }else{    Button_V4_create(OutOffScheme); }
    if(stan_V5){    Button_V5_create(OutOnScheme);    }else{    Button_V5_create(OutOffScheme); }

    if(stan_pompa1){Button_Pompa1_create(OutOnScheme); }else{    Button_Pompa1_create(OutOffScheme); }
    }
    if(stan_fan) {Button_Fan_create(OutOnScheme); }else{    Button_Fan_create(OutOffScheme); }
    }
    if(stan_O3){    Button_O3_create(OutOnScheme);    }else{    Button_O3_create(OutOffScheme); }
    }
    if(stan_PK1){    Button_PK1_create(OutOnScheme);    }else{    Button_PK1_create(OutOffScheme); }
    }
    if(stan_PK2){    Button_PK2_create(OutOnScheme);    }else{    Button_PK2_create(OutOffScheme); }
    }

    if(stan_red){    Button_R_create(OutOnScheme); }else{    Button_R_create(OutOffScheme); }
    if(stan_green){    Button_G_create(OutOnScheme); }else{    Button_G_create(OutOffScheme); }
    if(stan_blue){    Button_B_create(OutOnScheme); }else{    Button_B_create(OutOffScheme); }

    Button_Back_create();
}
//=====

//-----
WORD MsgEkranTesty(WORD objMsg, OBJ_HEADER* pObj){
    OBJ_HEADER* plnnyBtn; //zmienna potrzebna do blokowania klawiszy + i -
    switch(GetObjID(pObj)){
        //===== przyciski ===
        case ID_BUTTON_V1:
            if(objMsg == BTN_MSG_RELEASED){
                if(stan_V1){stan_V1=0;    ((BUTTON*)pObj)->hdr.pGolScheme = OutOffScheme;
            }
            else{
                stan_V1=1;    ((BUTTON*)pObj)->hdr.pGolScheme =
OutOnScheme;    }

```

```

    }
    return 1;
case ID_BUTTON_V2:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_V2){stan_V2=0; BtnSetScheme((BUTTON*)pObj,OutOffScheme); }
        else{ stan_V2=1; BtnSetScheme((BUTTON*)pObj,OutOnScheme);
    }
    }
    return 1;
case ID_BUTTON_V3:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_V3){stan_V3=0; ((BUTTON*)pObj)->hdr.pGolScheme = OutOffScheme;
    }
    else{ stan_V3=1; ((BUTTON*)pObj)->hdr.pGolScheme =
OutOnScheme; }
    }
    return 1;
case ID_BUTTON_V4:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_V4){stan_V4=0; BtnSetScheme((BUTTON*)pObj,OutOffScheme); }
        else{ stan_V4=1; BtnSetScheme((BUTTON*)pObj,OutOnScheme);
    }
    }
    return 1;
case ID_BUTTON_V5:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_V5){stan_V5=0; BtnSetScheme((BUTTON*)pObj,OutOffScheme); }
        else{ stan_V5=1; BtnSetScheme((BUTTON*)pObj,OutOnScheme);
    }
    }
    return 1;

case ID_BUTTON_POMPA1:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_pompa1){stan_pompa1=0; BtnSetScheme((BUTTON*)pObj,OutOffScheme);
    }
    else{ stan_pompa1=1;
BtnSetScheme((BUTTON*)pObj,OutOnScheme); }
    }
    return 1;
case ID_BUTTON_FAN:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_fan){ stan_fan=0;
BtnSetScheme((BUTTON*)pObj,OutOffScheme); }
        else{ stan_fan=1;
BtnSetScheme((BUTTON*)pObj,OutOnScheme); }
    }
    return 1;
case ID_BUTTON_PK1:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_PK1){ stan_PK1=0;
BtnSetScheme((BUTTON*)pObj,OutOffScheme); }
        else{ stan_PK1=1;
BtnSetScheme((BUTTON*)pObj,OutOnScheme); }
    }
    return 1;
case ID_BUTTON_PK2:
    if(objMsg == BTN_MSG_RELEASED){

```

```

        if(stan_PK2){      stan_PK2=0;
BtnSetScheme((BUTTON*)pObj, OutOffScheme);    }
        else{              stan_PK2=1;
BtnSetScheme((BUTTON*)pObj, OutOnScheme);      }
    }
    return 1;
case ID_BUTTON_O3:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_O3){      stan_O3=0;
BtnSetScheme((BUTTON*)pObj, OutOffScheme);    }
        else{              stan_O3=1;
BtnSetScheme((BUTTON*)pObj, OutOnScheme);      }
    }
    return 1;

case ID_BUTTON_RED:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_red){      stan_red=0;
BtnSetScheme((BUTTON*)pObj, OutOffScheme);    }
        else{              stan_red=1;
BtnSetScheme((BUTTON*)pObj, OutOnScheme);      }
    }
    return 1;
case ID_BUTTON_GREEN:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_green){    stan_green=0;
BtnSetScheme((BUTTON*)pObj, OutOffScheme);    }
        else{              stan_green=1;
BtnSetScheme((BUTTON*)pObj, OutOnScheme);      }
    }
    return 1;
case ID_BUTTON_BLUE:
    if(objMsg == BTN_MSG_RELEASED){
        if(stan_blue){     stan_blue=0;
BtnSetScheme((BUTTON*)pObj, OutOffScheme);    }
        else{              stan_blue=1;
BtnSetScheme((BUTTON*)pObj, OutOnScheme);      }
    }
    return 1;

case ID_BUTTON_BACK:
    if(objMsg == BTN_MSG_RELEASED){      ekran=CREATE_SERWIS;    }
    return 1;

default:
    return 1;                          // default
}
}
//=====
void CreateEkranMain(){
    GOLFree();
    if(strona_graf==1){      ZapisOkna2();} else { ZapisOkna1();}
    zmien_strone_graf=1;
    if(jezyk==ANGIELSKI){ PutImageExt(&ekr_main_AN, 0, 0);    //zawiera już bitmapy klawiszy
    }else{PutImageExt(&ekr_main_PL, 0, 0);}                      //POLSKI;
    rysuj_ikony=1;
    animacje_aktywne=1;
    SetColor(GREGWHITE);
    Bar(175,170,204,181);
    SetColor(BLACK);

```

```

    czas_sekundy_max=CZAS_ZABIEGU_MAX;
    czas_sekundy_min=CZAS_ZABIEGU_MIN;
    if(czas_zabiegu<czas_sekundy_min) czas_zabiegu=czas_sekundy_min;
    czas_sekundy=czas_zabiegu;      //czas bieżący = zapamiętany
    rysuj_czas_sekundy=1;    //przerysuj czas
                                // button ID, dimension ,radius, status, bitmap, text, color scheme//rekaw_upK
    BtnCreate(ID_BUTTON1, 10,56,58,109, 0,      BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
//zombek
    BtnCreate(ID_BUTTON2, 66,56,114,109, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
//kanal
    BtnCreate(ID_BUTTON3, 10,116,58,169, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
//butla
    BtnCreate(ID_BUTTON4, 66,116,114,169, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //szklanka
    BtnCreate(ID_BUTTON5, 10,174,114,227, 0,      BTN_DRAW|BTN_BMP_ONLY, NULL, NULL,
ProcScheme); //klucze (void*)&klucze,
    if(stezenie_ozonu==1){
        BtnCreate(ID_BUTTON_A, 135,82,163,110, 0, BTN_DRAW|BTN_BMP_ONLY, (void*)&a_wybrane, NULL,
ProcScheme); //stezenia_A
    }else{
        BtnCreate(ID_BUTTON_A, 135,82,163,110, 0, BTN_DRAW|BTN_BMP_ONLY, (void*)&a_stezenie, NULL,
ProcScheme); //stezenia_A
    }
    if(stezenie_ozonu==2){
        BtnCreate(ID_BUTTON_B, 167,82,195,110, 0, BTN_DRAW|BTN_BMP_ONLY, (void*)&b_wybrane, NULL,
ProcScheme); //stezenia_B
    }else{
        BtnCreate(ID_BUTTON_B, 167,82,195,110, 0, BTN_DRAW|BTN_BMP_ONLY, (void*)&b_stezenie, NULL,
ProcScheme); //stezenia_B
    }
    if(stezenie_ozonu==3){
        BtnCreate(ID_BUTTON_C, 199,82,227,110, 0, BTN_DRAW|BTN_BMP_ONLY, (void*)&c_wybrane, NULL,
ProcScheme); //stezenia_C
    }else{
        BtnCreate(ID_BUTTON_C, 199,82,227,110, 0, BTN_DRAW|BTN_BMP_ONLY, (void*)&c_stezenie, NULL,
ProcScheme); //stezenia_C
    }
    BtnCreate(ID_BUTTON_TMINUS, 135,190,163,218, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //czas minus
    BtnCreate(ID_BUTTON_TPLUS, 199,190,227,218, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //czas plus
    BtnCreate(ID_BUTTON_INFO, 271,191,299,219, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //info

//sterowanie wyjść
DiodyRGB_Off();
stan_pompa1=STAN_OFF;
    ZaworyOff();      //wyłączenie wszystkich zaworów
    stan_O3=STAN_OFF; //wyłączenie generatora
    errorokno=0;
}
//-----
WORD MsgEkranMain(WORD objMsg, OBJ_HEADER* pObj){
    //OBJ_HEADER* plnnyBtn;//zmienna potrzebna do..
    switch(GetObjID(pObj)){
        //===== przyciski ===
        case ID_BUTTON1:
            if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&zombek_szary8);}
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_ZOMBEK;}
    }

```



```

        return 1;
    case ID_BUTTON2:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&kanal_szary8);}
        if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_KANAL; }
        return 1;
    case ID_BUTTON3:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&butla_szary8);}
        if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_BUTLA; }
        return 1;
    case ID_BUTTON4:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&szklanka_szary8);}
        if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_SZKLANKA; }
        return 1;
    case ID_BUTTON5:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&kluce_szary8);}
        if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_CONFIG;}
        return 1;
    case ID_BUTTON_A:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&a_stezenie_szary8);}
        if(objMsg == BTN_MSG_RELEASED){
            stezenie_ozonu=1;
            BtnSetBitmap(pObj,(void*)&a_wybrane);
            PutImage(167,82, (void*)&b_stezenie,IMAGE_NORMAL);
            PutImage(199,82, (void*)&c_stezenie,IMAGE_NORMAL);
        }
        return 1; // pButon_A,
pButon_B, pButon_C, a_stezenie
    case ID_BUTTON_B:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&b_stezenie_szary8);}
        if(objMsg == BTN_MSG_RELEASED){
            stezenie_ozonu=2;
            BtnSetBitmap(pObj,(void*)&b_wybrane);
            PutImage(135,82, (void*)&a_stezenie,IMAGE_NORMAL);
            PutImage(199,82, (void*)&c_stezenie,IMAGE_NORMAL);
        }
        return 1;
    case ID_BUTTON_C:
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&c_stezenie_szary8);}
        if(objMsg == BTN_MSG_RELEASED){
            stezenie_ozonu=3;
            BtnSetBitmap(pObj,(void*)&c_wybrane);
            PutImage(135,82, (void*)&a_stezenie,IMAGE_NORMAL);
            PutImage(167,82, (void*)&b_stezenie,IMAGE_NORMAL);
        }
        return 1;
    case ID_BUTTON_INFO:
        if(objMsg == BTN_MSG_PRESSED){
            BtnSetBitmap(pObj,(void*)&info_szary8);
            SetColor(WHITE);
            SetFont( (void*)&GOLSmallFont );
            SetFontOrientation(1); // od dołu do góry
            OutTextXY(266,180,"Metrum CryoFlex"); //
            OutTextXY(286,180,"Made in Poland"); //
            SetFontOrientation(0); // poziom
        }
        if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_MAIN;}
        return 1;

//-----

```

```

case ID_BUTTON_TMINUS:
    if(objMsg == BTN_MSG_PRESSED){
        BtnSetBitmap(pObj,(void*)&minus_szary8);
        if(czas_sekundy>czas_sekundy_min){czas_sekundy--; rysuj_czas_sekundy=1; }

        TmrRepeatZero();
    }
    if(objMsg == BTN_MSG_STILLPRESSED){
        if(repeat_flag){
            if(TmrRepeat()>30){
                if(czas_sekundy>czas_sekundy_min){ czas_sekundy--;

                TmrRepeatZero();
            }
        }
        }else{
            if(TmrRepeat()>300){
                repeat_flag=1;
                TmrRepeatZero();
            }
        }
    }
    if(objMsg == BTN_MSG_RELEASED){
        repeat_flag=0;
        BtnSetBitmap(pObj,(void*)&minus);
    }
    if(objMsg == BTN_MSG_CANCELPRESS){
        BtnSetBitmap(pObj,(void*)&minus);
    }
    czas_zabiegu=czas_sekundy;    //zapamietaj wybrany czas jako czas zabiegu

    return 1;
//-----
case ID_BUTTON_TPLUS:
    if(objMsg == BTN_MSG_PRESSED){
        BtnSetBitmap(pObj,(void*)&plus_szary8);
        if(czas_sekundy<czas_sekundy_max){ czas_sekundy++; rysuj_czas_sekundy=1; }

        TmrRepeatZero();
    }
    if(objMsg == BTN_MSG_STILLPRESSED){
        if(repeat_flag){
            if(TmrRepeat()>30){

            if(czas_sekundy<czas_sekundy_max){czas_sekundy++;rysuj_czas_sekundy=1; }
                TmrRepeatZero();
            }
        }
        }else{ if(TmrRepeat()>300){ repeat_flag=1; TmrRepeatZero(); }
    }

    if(objMsg == BTN_MSG_RELEASED){
        repeat_flag=0;
        BtnSetBitmap(pObj,(void*)&plus);
    }
    if(objMsg == BTN_MSG_CANCELPRESS){
        BtnSetBitmap(pObj,(void*)&plus);
    }
    czas_zabiegu=czas_sekundy;    //zapamietaj wybrany czas jako czas zabiegu
    return 1;

```



```

//-----

default:
    return 1;                                // default
}
}
void BlueOkno(void){
    SetColor(LIGHTBLUE);
    Bar(20,60,300,170);
    SetColor(TLO_NR3);
    Arc(23,63,297,167,4,5,30);
    SetColor(TLO_NR2);
    Arc(23,63,297,167,4,5,225);
    SetColor(TLO_NR2);
    Arc(23,63,297,167,6,7,30);
    SetColor(TLO_NR3);
    Arc(23,63,297,167,6,7,225);
}
void ZmianyEkranuMain(){
    //polewanie wodą naozonowaną - możliwe gdy
    static char polewanie=0;
    if(nogal_blue){
        switch(stan_butli){
            case BUTLA_O3_BRAK:
                if(errorokno==0){
                    errorokno=1;    //komunikat - brak odpowiedniej wody
                    BlueOkno();
                    SetColor(BLACK);
                    SetFont((void*)&GOLFontDefault);
                    OutTextXY(30,75,TextSel(język,Pauza1));
                    OutTextXY(30,120,TextSel(język,Pauza2));
                }
                if((polewanie==1)|| (polewanie==2)) {polewanie=3; ZeroCzasEtapu();} //a tu
przerwanie polewania, wymuszone zmianą stanu butli
                break;
            case BUTLA_O3_GRANICA:    //np: dźwięk ostrzegający
            case BUTLA_O3_OK:
                //woda dobra, lub ujdzie: polewanie sterowane przez nogal_blue
                if(polewanie==0){polewanie=1; ZeroCzasEtapu();}
                break;
        }
    }
    }else{
        if((polewanie==1)|| (polewanie==2)) {polewanie=3; ZeroCzasEtapu();}    //było zaczęte
polewanie, teraz kończę
        if((errorokno==1)&&(polewanie==0)) {ekran=CREATE_MAIN; errorokno=0;}
    }
    //sterowanie polewaniem
    switch(polewanie){
        case 0: stan_V2=STAN_OFF; stan_V4=STAN_OFF; stan_V3=STAN_OFF;
        break;
        case 1: stan_V2=STAN_ON; stan_V4=STAN_OFF; stan_V3=STAN_OFF;
        if(CzasEtapuCs()==100){polewanie=2; } //przygotowanie polewania
        break;
        case 2: stan_V2=STAN_ON; stan_V4=STAN_ON; stan_V3=STAN_OFF; //polewanie właściwe
        break;
        case 3: stan_V2=STAN_OFF; stan_V4=STAN_ON; stan_V3=STAN_ON;
        if(CzasEtapuCs()==100){polewanie=4; } //włączam V4 na 2 sek, spust ciśnienia
        break;
    }
}

```

```

        case 4: stan_V2=STAN_OFF; stan_V4=STAN_OFF; stan_V3=STAN_ON;
if(CzasEtapuCs()==200){polewanie=0; }
        break;
        default: polewanie=0;
        break;
    }
}
void WeryfikujKonfiguracje(){
    if((konfiguracja_aparatu[USER_CZAS] != czas_zabiegu) || (konfiguracja_aparatu[USER_POZIOM] !=
stezenie_ozonu)) ZapisKonfiguracji();
}
//=====
void CreateEkranZombek(){
    GOLFree();
    if(strona_graf==1){ ZapisOkna2();} else { ZapisOkna1();}
    zmien_strone_graf=1;
    if(jezyk==ANGIELSKI){ PutImageExt(&ekr_zabieg_AN, 0, 0); //zawiera już bitmapy klawiszy
    }else{PutImageExt(&ekr_zabieg_PL, 0, 0);} //POLSKI;
    rysuj_ikony=1;
    animacje_aktywne=1;
    SetColor(GREGWHITE);
    Bar(175,170,204,181); //kasowanie "min./"
    //PutImageExt(&ekr_zabieg, 0, 0); //zawiera już bitmapy klawiszy
    PutImage(10,56, (void*)&ico_zomb8,IMAGE_NORMAL);
    WeryfikujKonfiguracje(); //sprawdza i zapisuje w pamięci ewentualne modyfikacje konfiguracji
    SetColor(BLACK);
    if(stezenie_ozonu==1){ PutImage(135,82, (void*)&a_wybrane,IMAGE_NORMAL); }
    if(stezenie_ozonu==2){ PutImage(167,82, (void*)&b_wybrane,IMAGE_NORMAL); }
    if(stezenie_ozonu==3){ PutImage(199,82, (void*)&c_wybrane,IMAGE_NORMAL); }
    stezenie_robocze=stezenie_ozonu; //
    BtnCreate(ID_BUTTON_STOP, 250,89,319,187, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //STOP-czerwony w ekranie
    czas_zabiegu=czas_sekundy; //zapamiętaj wybrany czas jako czas zabiegu
    //SetCzasKoncaZabiegu(czas_sekundy);
    rysuj_czas_sekundy=1;
    //sterowanie wyjść
    ZaworyOff(); //wyłączenie wszystkich zaworów
    stan_V3=STAN_ON; //za wyjątkiem V4
    stan_O3=STAN_OFF; //wyłączenie generatora
    UstawStezenieRobocze(stezenie_robocze);
    blokada=0;
    stan_zabiegu=0;
    stan_pauzy=0;
}
//-----
WORD MsgEkranuZombek(WORD objMsg, OBJ_HEADER* pObj){
    if(GetObjID(pObj)==ID_BUTTON_STOP){
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&stop_szary8);}
        if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_MAIN;}
    }
    return 1;
}
void ZmianyEkranuZombek(){
    WORD czas_do_konca;
    if(!stan_pauzy && stan_zabiegu && FlagaZmianyCzasuKonca()){ //tylko gdy stan_zabiegu={1,2,3}
        KasujFlageZmianyCzasuKonca();
        czas_sekundy = GetCzasKoncaZabiegu(); //uaktualnienie wyświetlanego czasu
        rysuj_czas_sekundy=1; //ustawienie flagi przerysowania
    }
}

```

```

if(nogal_yellow){DiodyRGB_On();}else{DiodyRGB_Off();}; //niezależna sygnalizacja diodami, kolor - stężenie,
ie - naciśnięcie
if(blokada){
    if(CzasEtapu())>=CZAS_BLOKADY){ekran=CREATE_MAIN;}
    if(CzasEtapuCs())>50){stan_V1=STAN_OFF; }
    SetColor(LIGHTBLUE);
    Bar(60,234,60+(2*CzasEtapuCs())/CZAS_BLOKADY),238); //sygnalizacja upływu czasu blokady
}else{
    if(nogal_yellow){
        if(stan_pauzy){
            stan_pauzy=0; //tu
            //stan_V2=STAN_OFF;
            if(czas_pauza > 7){ //zostało
                stan_zabiegu=1; //ponawiamy spust
                ZeroCzasEtapu(); //czasy dla faz
            }
            SetCzasKoncaZabiegu(czas_pauza); //restartujemy zabieg od czasu
            //czyszczenie progres-baru pauzy, kolorami pasującymi do tła
            SetColor(RGB565CONVERT(142, 218, 41)); Bar(60,234,260,236);
            SetColor(RGB565CONVERT(121, 204, 11)); Line(60,237,260,237);
            SetColor(RGB565CONVERT(103, 179, 5)); Line(60,238,260,238);
        }
        czas_do_konca=GetCzasKoncaZabiegu(); //aktualizacja czasu zabiegu, dla stan_zabiegu=1 i
        switch(stan_zabiegu){
            //stan 0, startujemy zabieg od zera
            case 0: czas_do_konca=czas_zabiegu; SetCzasKoncaZabiegu(czas_zabiegu);
            case 1: stan_pompa1=STAN_ON; stan_V1=STAN_OFF; stan_V2=STAN_ON;
            case 2: stan_pompa1=STAN_ON; stan_V1=STAN_OFF; stan_V2=STAN_OFF;
            case 3: stan_pompa1=STAN_ON; stan_V1=STAN_ON; stan_V2=STAN_OFF;
            case 4: stan_pompa1=STAN_ON; stan_V1=STAN_ON; stan_V2=STAN_OFF;
            case 5: stan_pompa1=STAN_ON; stan_V1=STAN_ON; stan_V2=STAN_OFF;
        }
        if(stan_O3==STAN_OFF){rysuj_piorun=1;}
        stan_O3=STAN_ON;
        if(czas_do_konca < 7){stan_zabiegu=5; };
        if(!czas_do_konca ){
            //stan .., koniec, blokada
        }
    }
}

```

```

        stan_pompa1=STAN_OFF;
        //tylko pompa_OFF, V1 zamykam w czasie blokady
        blokada=1;
        //flaga blokady
        SetColor(LIGHTBLUE); Line(60,236,260,236);

//podkład pod sygnalizacje upływu blokady
        ZeroCzasEtapu();
        //start liczenia czasu blokady
        };
        break;
    }
}
}else{
    if(stan_zabiegu){ //pauza nie możliwa przed rozpoczęciem zabiegu
        switch(stan_pauzy){
            // 0, ustawienie pauzy
            case 0: czas_pauza=GetCzasKoncaZabiegu(); ZeroCzasEtapu(); stan_pauzy=1;
                    stan_O3=STAN_OFF; stan_V1=STAN_OFF;

stan_V2=STAN_OFF;

                    rysuj_piorun=1;
                    SetColor(YELLOW); Line(60,236,260,236); //podkład pod
sygnalizacje upływu pauzy

                    //pauza trwa
                    case 1: SetColor(YELLOW);
Bar(60,234,60+(2*CzasEtapuCs())/CZAS_PAUZY,238); //sygnalizacja upływu pauzy
                    if(CzasEtapu()>=CZAS_PAUZY){ekran=CREATE_MAIN;
stan_pompa1=STAN_OFF; stan_V2=STAN_OFF; };
                    break;
                }
            }
        }
    }
}

//=====
void CreateEkranKanal(){
    GOLFree();
    if(strona_graf==1){ ZapisOkna2();} else { ZapisOkna1();}
    zmien_strone_graf=1;
    if(jezyk==ANGIELSKI){ PutImageExt(&ekr_zabieg_AN, 0, 0); //zawiera już bitmapy klawiszy
    }else{PutImageExt(&ekr_zabieg_PL, 0, 0);} //POLSKI;
    rysuj_ikony=1;
    animacje_aktywne=1;
    SetColor(GREGWHITE);
    Bar(175,170,204,181); //kasowanie "min./"
    PutImage(10,56, (void*)&ico_kanal8,IMAGE_NORMAL);
    WeryfikujKonfiguracje(); //sprawdza i zapisuje w pamięci ewentualne modyfikacje konfiguracji
    SetColor(BLACK);
    if(stezenie_ozonu==1){ PutImage(135,82, (void*)&a_wybrane,IMAGE_NORMAL); }
    if(stezenie_ozonu==2){ PutImage(167,82, (void*)&b_wybrane,IMAGE_NORMAL); }
    if(stezenie_ozonu==3){ PutImage(199,82, (void*)&c_wybrane,IMAGE_NORMAL); }
    stezenie_robocze=stezenie_ozonu; //
    BtnCreate(ID_BUTTON_STOP, 250,89,319,187, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //STOP-czerwony w ekranie
    czas_zabiegu=czas_sekundy; //zapamiętaj wybrany czas jako czas zabiegu
    SetCzasKoncaZabiegu(czas_zabiegu);
    rysuj_czas_sekundy=1;
    //sterowanie wyjść
    ZaworyOff(); //wyłączenie wszystkich zaworów
    stan_V3=STAN_ON; //za wyjątkiem V4
    stan_O3=STAN_OFF; //wyłączenie generatora

```

```

    UstawStezenieRobocze(stezenie_robocze);
    blokada=0;
    stan_zabiegu=0;
    stan_pauzy=0;
}
//-----
WORD MsgEkranuKanal(WORD objMsg, OBJ_HEADER* pObj){
    if(GetObjID(pObj)==ID_BUTTON_STOP){
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&stop_szary8); }
        if(objMsg == BTN_MSG_RELEASED){      ekran=CREATE_MAIN;}
    }
    return 1;
}
//-----
void ZmianyEkranuKanal(){
    WORD czas_do_konca;
    if(!stan_pauzy && stan_zabiegu && FlagaZmianyCzasuKonca()){ //tylko gdy stan_zabiegu={1,2,3}
        KasujFlageZmianyCzasuKonca();
        czas_sekundy = GetCzasKoncaZabiegu(); //uaktualnienie wyświetlanego czasu
        rysuj_czas_sekundy=1; //ustawienie flagi przerysowania
    }
    if(nogal_yellow){DiodyRGB_On();}else{DiodyRGB_Off();}; //niezależna sygnalizacja diodami, kolor - stężenie,
    świecenie - naciśnięcie
    if(blokada){
        if(CzasEtapu())>=CZAS_BLOKADY){ekran=CREATE_MAIN;}
        if(CzasEtapuCs())>50){stan_V1=STAN_OFF; } //tu dopiero całkowite zakończenie zabiegu
        SetColor(LIGHTBLUE);
        Bar(60,234,60+(2*CzasEtapuCs())/CZAS_BLOKADY,238); //sygnalizacja upływu czasu blokady
    }else{
        if(nogal_yellow){
            if(stan_pauzy){
                stan_pauzy=0; //tu
                wychodzimy z pauzy
                if(czas_pauza > 7){ //zostało
                    jeszcze 7 sekund
                    stan_zabiegu=1; //ponawiamy spust
                    nadciśnienia
                    ZeroCzasEtapu(); //czasy dla faz
                    spustu
                }
                SetCzasKoncaZabiegu(czas_pauza); //restartujemy zabieg od czasu
                zapamiętanego na początku pauzy
                //czyszczenie progres-baru pauzy, kolorami pasującymi do tła
                SetColor(RGB565CONVERT(142, 218, 41)); Bar(60,234,260,236);
                SetColor(RGB565CONVERT(121, 204, 11)); Line(60,237,260,237);
                SetColor(RGB565CONVERT(103, 179, 5)); Line(60,238,260,238);
            }
            czas_do_konca=GetCzasKoncaZabiegu(); //aktualizacja czasu zabiegu, dla stan_zabiegu=1 i
            wyższych.
            switch(stan_zabiegu){
                //stan 0, startujemy zabieg od zera
                case 0: czas_do_konca=czas_zabiegu; SetCzasKoncaZabiegu(czas_zabiegu);
                stan_zabiegu=1; ZeroCzasEtapu();
                case 1: stan_pompa1=STAN_ON; stan_V1=STAN_OFF; stan_V2=STAN_ON;
                stan_V3=STAN_ON;
                if(CzasEtapuCs())>80){stan_zabiegu=2; };
                break;
                case 2: stan_pompa1=STAN_ON; stan_V1=STAN_OFF; stan_V2=STAN_OFF;
                if(CzasEtapuCs())>100){stan_zabiegu=3; };

```

```

        break;
        case 3: stan_pompa1=STAN_ON; stan_V1=STAN_ON; stan_V2=STAN_OFF;
                if(czas_do_konca < czas_zabiegu-5){stan_zabiegu=4; }; //stan
        .., minęło już 5 sekund

        break;
        case 4: stan_pompa1=STAN_ON; stan_V1=STAN_ON; stan_V2=STAN_OFF; //stan
        "PODSTAWOWY" zabiegu

                if(stan_O3==STAN_OFF){rysuj_piorun=1;}

        ///!! OPERACJA JEDNORAZOWA

                stan_O3=STAN_ON;
                if(czas_do_konca < 7){stan_zabiegu=5; };

        //stan .., zostało jeszcze 7 sekund

        break;
        case 5: stan_pompa1=STAN_ON; stan_V1=STAN_ON; stan_V2=STAN_OFF;
                if(stan_O3==STAN_ON){rysuj_piorun=1;}

        ///!! OPERACJA JEDNORAZOWA

                stan_O3=STAN_OFF;
                if(!czas_do_konca ){

        //stan .., koniec, blokada

                stan_pompa1=STAN_OFF;
                //tylko pompa_OFF, V1 zamykam w czasie blokady
                blokada=1;
                //flaga blokady
                SetColor(LIGHTBLUE); Line(60,236,260,236);

        //podkład pod sygnalizacje upływu blokady

                ZeroCzasEtapu();

                //start liczenia czasu blokady
                };

        break;

        }

    }else{

        if(stan_zabiegu){ //pauza nie możliwa przed rozpoczęciem zabiegu
            switch(stan_pauzy){
                // 0, ustawienie pauzy
                case 0: czas_pauza=GetCzasKoncaZabiegu(); ZeroCzasEtapu(); stan_pauzy=1;
                        stan_O3=STAN_OFF; stan_V1=STAN_OFF;

        stan_V2=STAN_OFF;

                        rysuj_piorun=1;
                        SetColor(YELLOW); Line(60,236,260,236); //podkład pod
        sygnalizacje upływu pauzy

                        //pauza trwa

                case 1: SetColor(YELLOW);
        Bar(60,234,60+(2*CzasEtapuCs())/CZAS_PAUZY,238); //sygnalizacja upływu pauzy
                        if(CzasEtapu()>=CZAS_PAUZY){ekran=CREATE_MAIN;

        stan_pompa1=STAN_OFF; stan_V2=STAN_OFF; };
                        break;

                }

            }

        }

    }

}

//=====
void CreateEkranButla(){
    GOLFree();
    if(strona_graf==1){ ZapisOkna2(); } else { ZapisOkna1(); }
    zmien_strone_graf=1;
    if(jezyk==ANGIELSKI){ PutImageExt(&ekr_zabieg_AN, 0, 0); //zawiera już bitmapy klawiszy
    }else{PutImageExt(&ekr_zabieg_PL, 0, 0); } //POLSKI;
    PutImage(10,56, (void*)&ico_butla8,IMAGE_NORMAL);
}

```



```

rysuj_ikony=1;
animacje_aktywne=1;
SetColor(BLACK);
stezenie_robocze=3;
PutImage(199,82, (void*)&c_wybrane,IMAGE_NORMAL);
czas_sekundy=czas_ozonowania_butli;           //sekundy CZAS_BUTLI;
SetCzasKoncaZabiegu(czas_sekundy);
char min,sek,dsek,jsek;
min=czas_sekundy/60;
sek=czas_sekundy%60;
dsek=sek/10;
jsek=sek%10;
DisplayCzas_9_59(min,dsek,jsek,SZARA_CYFRA);
BtnCreate(ID_BUTTON_STOP, 250,89,319,187, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
//STOP-czerwony w ekranie
//sterowanie wyjść
ZaworyOff();           //wyłączenie wszystkich zaworów
stan_O3=STAN_OFF;      //wyłączenie generatora
stan_V2=STAN_ON ;
stan_V3=STAN_ON ;
UstawStezenieRobocze(stezenie_robocze);
ZeroCzasEtapu();       //sterowanie etapami
DiodyRGB_On();
}
WORD MsgEkranuButla(WORD objMsg, OBJ_HEADER* pObj){
    if(GetObjID(pObj)==ID_BUTTON_STOP){
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&stop_szary8); }
        if(objMsg == BTN_MSG_RELEASED){     ekran=CREATE_MAIN;}
    }
    return 1;
}
void ZmianyEkranuButla(){
    static char pop_min,pop_dsek,pop_jsek;
    char min,sek,dsek,jsek;
    WORD czas;
    if(FlagaZmianyCzasuKonca() ){
        KasujFlageZmianyCzasuKonca();
        czas = GetCzasKoncaZabiegu();    //uaktualnienie wyświetlanego czasu
        min=czas/60;
        sek=czas%60;
        dsek=sek/10;
        jsek=sek%10;
        DisplayCzas_9_59(pop_min,pop_dsek,pop_jsek,GREGWHITE);
        DisplayCzas_9_59(min,dsek,jsek,SZARA_CYFRA);
        pop_min=min;
        pop_dsek=dsek;
        pop_jsek=jsek;
        if(!czas){           //koniec
            ekran=CREATE_MAIN;
            stan_butli = BUTLA_O3_OK;
            ZeroTmrButli();
        }
    }
    if((CzasEtapu()==5)&&(stan_O3==STAN_OFF)){           //!!! OPERACJA
JEDNORAZOWA
        stan_O3=STAN_ON;
        rysuj_piorun=1;
    }
    if((CzasEtapu()==(czas_ozonowania_butli-5))&&(stan_O3==STAN_ON)){           //!!! OPERACJA JEDNORAZOWA

```



```

        //zamiana na jednorazową, np:      if((CzasEtapu()==(czas_ozonowania_butli-
5))&&(stan_O3==STAN_ON))
        stan_O3=STAN_OFF;
        rysuj_piorun=1;
    }
}
//=====
void CreateEkranSzkłanka(){
    GOLFree();
    if(strona_graf==1){      ZapisOkna2();} else { ZapisOkna1();}
    zmien_strone_graf=1;
    if(jezyk==ANGIELSKI){ PutImageExt(&ekr_zabieg_AN, 0, 0);      //zawiera już bitmapy klawiszy
    }else{PutImageExt(&ekr_zabieg_PL, 0, 0);}                        //POLSKI;
    rysuj_ikony=1;
    animacje_aktywne=1;
    SetColor(GREGWHITE);
    Bar(175,170,204,181);      //kasowanie "min./"
    Bar(133,62,228,74);      //kasowanie "STĘŻENIE"
    PutImage(10,56, (void*)&ico_szkłanka8,IMAGE_NORMAL);
    BtnCreate(ID_BUTTON_STOP, 250,89,319,187, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //STOP-czerwony w ekranie
    czas_sekundy=10;
    SetCzasKoncaZabiegu(czas_sekundy);
    rysuj_czas_sekundy=1;
    //sterowanie wyjść
    ZaworyOff();      //wyłączenie wszystkich zaworów
    stan_O3=STAN_OFF;      //wyłączenie generatora
    errorokno=0;
}
WORD MsgEkranuSzkłanka(WORD objMsg, OBJ_HEADER* pObj){
    if(GetObjID(pObj)==ID_BUTTON_STOP){
        if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&stop_szary8); }
        if(objMsg == BTN_MSG_RELEASED){      ekran=CREATE_MAIN; stan_V5=STAN_OFF ;}
    }
    return 1;
}
void ZmianyEkranuSzkłanka(){
    WORD czas;
    if((stan_butli==BUTLA_O3_BRAK)&&(stan_V5==STAN_OFF)){
        ekran=CREATE_OKIENKO;
        errorokno=1;      //komunikat - brak odpowiedniej wody
    }else{
        stan_V2=STAN_ON;
        stan_V5=STAN_ON ;
        if(FlagaZmianyCzasuKonca() ){
            KasujFlageZmianyCzasuKonca();
            czas_sekundy = GetCzasKoncaZabiegu();      //uaktualnienie wyświetlanego czasu
            rysuj_czas_sekundy=1;                        //ustawienie flagi przerysowania
            if(!czas_sekundy){      //koniec
                ekran=CREATE_MAIN;
                stan_V2=STAN_OFF;
                stan_V3=STAN_OFF;
                stan_V5=STAN_OFF ;
            }
            if(czas_sekundy==2) stan_V3=STAN_ON;
        }
    }
}
}

```

```
//=====
void CreateEkranOkienko(void){
    GOLFree();
    BlueOkno();    //grafika lub bitmapa - podkład
    switch(errorokno){
        case 1:
            SetColor(BLACK);
            SetFont((void*)&GOLFontDefault);
            OutTextXY(30,75,TextSel(jezyk,Pauza1));
            OutTextXY(30,120,TextSel(jezyk,Pauza2));
        }
    BtnCreate(ID_BUTTON_OKNO, 20,60,300,170, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //STOP-czerwony w ekranie

    SetCzasKoncaZabiegu(10);
    ZaworyOff();    //wyłączenie wszystkich zaworów
    stan_O3=STAN_OFF;    //wyłączenie generatora
}
WORD MsgEkranuOkienko(WORD objMsg, OBJ_HEADER* pObj){
    if(GetObjID(pObj)==ID_BUTTON_OKNO){
        if(objMsg == BTN_MSG_PRESSED){ }
        if(objMsg == BTN_MSG_RELEASED){    ekran=CREATE_MAIN;}
    }
    return 1;
}
void ZmianyEkranuOkienko(void){
    WORD czas;
    if(!GetCzasKoncaZabiegu()){ekran=CREATE_MAIN;}; //koniec ||;
}
//=====
typedef enum {
    KONFIG_START=0,
    KONFIG_JEZYK,
    KONFIG_LAMPA,
    KONFIG_BUZIA,
    KONFIG_NUTA,
} KONFIG_FAZY;
KONFIG_FAZY konfig_stan = KONFIG_START; // aktualny statn ekranu info

void DispSuwakV(SHORT x,SHORT y1,SHORT y2,WORD zakres,WORD poz){
    //WORD i,k,m;
    //m=y2-y1;    //odwrotna względem zmiany Y, top i spód po 4 pixele
    //k=(poz*m)/zakres;
    //PutImage(x,y2+1, (void*)&suwak_spod_V,IMAGE_NORMAL);    //spód suwaka, wysokość=4
    //SetColor(GREEN);
    //for(i=0;i<k;i++)PutImage(x,y2-i, (void*)&suwak_line_V,IMAGE_NORMAL);    //kreski suwaka
    //SetColor(WHITE);
    //for(i=k;i<=m+4;i++){Line(x,y2-i,x+12,y2-i);} //wybielenie reszty suwaka
    //PutImage(x,y2-k-3, (void*)&suwak_top_V,IMAGE_NORMAL); //top suwaka, wysokość=4
}

void DrawSuwak(int kolor,SHORT x1,SHORT y1,SHORT x2,SHORT y2,WORD zakres,WORD poz){
    WORD i,k,m;
    m=y2-y1;
    k=(poz*m)/zakres;
    SetColor(kolor);
    for(i=0;i<k;i++){Line(x1,y2-i,x2,y2-i);}
    SetColor(WHITE);
}
```

```

        for(i=k;i<=m;i++){Line(x1,y2-i,x2,y2-i);}
    }
    /*
void DrawSuwak(SHORT x1,SHORT y1,SHORT x2,SHORT y2,WORD zakres,WORD poz){
    WORD i,k,m;
    m=x2-x1;
    k=(poz*m)/zakres;
    SetColor(GREEN);
    for(i=0;i<k;i++){Line(x1+i,y1,x1+i,y2);}
    SetColor(WHITE);
    for(i=k;i<=m;i++){Line(x1+i,y1,x1+i,y2);}
}
*/
void SkalaSuwaka(void){
    SetColor(RGB565CONVERT(120,120,120) );
    Line(195,80,195,200);
    Line(192,200,196,200);
    Line(192,80,196,80);
    //--
    Line(194,110,195,110);
    Line(194,140,195,140);
    Line(194,170,195,170);
}

void CreateEkranConfig(){

    GOLFree();
    //wybór strony graficznej.
    if(strona_graf==1){ ZapisOkna2();} else { ZapisOkna1();}
    zmien_strone_graf=1;
    PutImageExt(&ekr_konfig, 0, 0); //
    rysuj_ikony=1;
    animacje_aktywne=1;
    SetColor(BLACK);

    PutImage( 90,75, (void*)&flaga_PL8,IMAGE_NORMAL);
    PutImage(170,75, (void*)&flaga_GB8,IMAGE_NORMAL);
    // button ID, dimension, radius, status, bitmap, text,
    color scheme//
    BtnCreate(ID_BUTTON1, 85, 73,132, 98, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL,
    ProcScheme); //PL
    BtnCreate(ID_BUTTON2, 165, 73,212, 98, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL,
    ProcScheme); //GB
    BtnCreate(ID_BUTTON_MAIN, 250, 88,319,187, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL,
    ProcScheme); //(void*)&powrot_zoltyK

    //-----
    BtnCreate(ID_BUTTON_T1, 20,190, 50,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "1", ProcScheme);
    //1 min
    BtnCreate(ID_BUTTON_T2, 80,190,110,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "4",
    ProcScheme); //4 min
    BtnCreate(ID_BUTTON_T3, 140,190,170,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "7", ProcScheme);
    //7 min
    BtnCreate(ID_BUTTON_T4, 200,190,230,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "10", ProcScheme);
    //10 min
    //-----

    //jasność
    pSlider=SldCreate( ID_SLIDER1, 80, 130, 220, 150, SLD_DRAW, 39, 10, 0, ProcScheme);//SLD_DRAW
    /SLD_SCROLLBAR

```

```

//SldSetPos(pSlider,disp_jasnosc_led-1);           //zapobiega migotaniu
SldSetPos(pSlider,jasnosc_led-1);                 //zapobiega migotaniu
rys_jasnosc_led=1;
//UstawJasnosc(jasnosc_led);

rys_jezyk_flaga=1;                               //przerysowanie ramki/ramek wokół flag, CREATE odp przycisków

//sterowanie wyjść
ZaworyOff();                                     //wyłączenie wszystkich zaworów
stan_O3=STAN_OFF;                               //wyłączenie generatora
}
//-----

WORD MsgEkranConfig(WORD objMsg, OBJ_HEADER* pObj){
    unsigned char pom;
    switch(GetObjID(pObj)){
        case ID_SLIDER1:
            if((objMsg == SLD_MSG_INC) || (objMsg == SLD_MSG_DEC)){
                jasnosc_led=SldGetPos(pSlider)+1;
                rys_jasnosc_led=1;
                UstawJasnosc(jasnosc_led);
            }
            return 1;
        case ID_BUTTON1:                             //wybierz PL
            if(objMsg == BTN_MSG_PRESSED){
                if(jezyk!=POLSKI){
                    jezyk=POLSKI;
                    konfiguracja_aparatu[WYBRANY_JEZYK]=jezyk;
                    rys_jezyk_flaga=1;
                }
            }
            return 1;
        case ID_BUTTON2:                             //wybierz GB
            if(objMsg == BTN_MSG_PRESSED){
                if(jezyk!=ANGIELSKI){
                    jezyk=ANGIELSKI;
                    konfiguracja_aparatu[WYBRANY_JEZYK]=jezyk;
                    rys_jezyk_flaga=1;
                }
            }
            return 1;
        case ID_BUTTON_T1:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =60; }
            return 1;
        case ID_BUTTON_T2:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =240; }
            return 1;
        case ID_BUTTON_T3:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =420; }
            return 1;
        case ID_BUTTON_T4:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =599; }
            return 1;

        case ID_BUTTON_MAIN:
            if(objMsg == BTN_MSG_PRESSED) { BtnSetBitmap(pObj,(void*)&powrot_szary8);
            konfig_stan = KONFIG_START; }
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_MAIN; ZapisKonfiguracji();}
    }
}

```

```

        //zapis konfiguracji po wyjściu z ekranu
        return 1;

    default:
        return 1;
    }
}

// default

/*
void CreateEkranConfig(){
    GOLFree();
    //wybór strony graficznej.
    if(strona_graf==1){ ZapisOkna2();} else { ZapisOkna1();}
    zmien_strone_graf=1;
    PutImageExt(&ekr_konfig, 0, 0);    //
    rysuj_ikony=1;
    animacje_aktywne=1;
    SetColor(BLACK);

    // button ID, dimension ,radius, status , bitmap, text, color scheme//

    BtnCreate(ID_BUTTON1,      16,68,107,98, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //jezyk
    BtnCreate(ID_BUTTON2,      16,107,107,137, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //jasność
    BtnCreate(ID_BUTTON3,      16,146,107,176, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //Abc
    BtnCreate(ID_BUTTON4,      16,185,107,215, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //nutki
    BtnCreate(ID_BUTTON5,      271,191,301,221, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //info
    //-----
    BtnCreate(ID_BUTTON_T1,     10,190,30,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "1", ProcScheme);
    //1 min
    BtnCreate(ID_BUTTON_T2,     40,190,60,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "4", ProcScheme);
    //4 min
    BtnCreate(ID_BUTTON_T3,     70,190,90,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "7", ProcScheme);
    //7 min
    BtnCreate(ID_BUTTON_T4,     100,190,120,220, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, "10", ProcScheme);
    //10 min
    //-----
    BtnCreate(ID_BUTTON_MAIN,   250,88,319,187, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
    //(void*)&powrot_zoltyk

    switch(konfig_stan){
        case KONFIG_JEZYK:
            //PutImage(16,68, (void*)&jezyk_sel,IMAGE_NORMAL);    //wskazuje wybrany klawisz
            switch(jezyk){
                case POLSKI:
                    PutImage(153,90, (void*)&flaga_tlo_sel8,IMAGE_NORMAL);
                    PutImage(158,95, (void*)&flaga_PL8,IMAGE_NORMAL);
                    PutImage(158,145, (void*)&flaga_GB8,IMAGE_NORMAL);
                    BtnCreate(ID_BUTTON7, 153,140,205,172, 0, BTN_DRAW|BTN_BMP_ONLY, NULL, NULL, ProcScheme);
                    //GB
                    break;
                case ANGIELSKI:

```

```

        PutImage(153,140, (void*)&flaga_tlo_sel8,IMAGE_NORMAL);
        PutImage(158,95, (void*)&flaga_PL8,IMAGE_NORMAL);
        PutImage(158,145, (void*)&flaga_GB8,IMAGE_NORMAL);
        BtnCreate(ID_BUTTON6, 153,90,205,122, 0,  BTN_DRAW|BTN_BMP_ONLY, NULL,
        NULL, ProcScheme); //PL
        break;
    }
    break;
    case KONFIG_LAMPA:
        //PutImage(16,107, (void*)&jasnosc_sel,IMAGE_NORMAL); //wskazuje wybrany klawisz
        rys_jasnosc_led=1; //wstępne wyrysowanie
        pSlider=SldCreate( ID_SLIDER1, 155, 70, 200, 190, SLD_VERTICAL|SLD_BMP_ONLY, 39, 10, 0,
        ProcScheme);//SLD_DRAW |SLD_SCROLLBAR
        SkalaSuwaka();
        break;
    case KONFIG_NUTA:
        //PutImage(16,146, (void*)&vol_nuta_sel,IMAGE_NORMAL); //wskazuje wybrany klawisz
        rys_volume_dzwiek=1; //wstępne wyrysowanie
        pSlider=SldCreate( ID_SLIDER2, 155, 70, 200, 190, SLD_VERTICAL|SLD_BMP_ONLY, 40, 10, 0,
        ProcScheme);//SLD_DRAW |SLD_SCROLLBAR
        SkalaSuwaka();
        break;
    }
    //sterowanie wyjść
    ZaworyOff(); //wyłączenie wszystkich zaworów
    stan_O3=STAN_OFF; //wyłączenie generatora
}
//-----
void ZmianyEkranuConfig(){
    SldGetPos(pSlider);
}
void UstawVolume(char volume){
    unsigned char vol;
    if(volume<3){vol = 254;} //najmniejsze głośności traktuje jako wyłączenie dźwięku
    else{
        if(volume<41){vol = 130-3*volume;}
        else{vol=23;}
    }
    VS1011_SetVolume(vol);
}
WORD MsgEkranConfig(WORD objMsg, OBJ_HEADER* pObj){
    switch(GetObjID(pObj)){
        case ID_SLIDER1: //zmiany jasności podświetlenia
            if(objMsg == SLD_MSG_INC){jasnosc_led=SldGetPos(pSlider)+1; rys_jasnosc_led=1;
            UstawJasnosc(jasnosc_led); }
            if(objMsg == SLD_MSG_DEC){jasnosc_led=SldGetPos(pSlider)+1; rys_jasnosc_led=1;
            UstawJasnosc(jasnosc_led); }
            return 1;
        //case ID_SLIDER2: //zmiany głośności dźwięków
        // if(objMsg == SLD_MSG_INC){volume_dzwiek=SldGetPos(pSlider); rys_volume_dzwiek=1;
        UstawVolume(volume_dzwiek); if(!GramTeraz())Klik(); }
        // if(objMsg == SLD_MSG_DEC){volume_dzwiek=SldGetPos(pSlider); rys_volume_dzwiek=1;
        UstawVolume(volume_dzwiek); if(!GramTeraz())Klik(); }
        // return 1;
        case ID_BUTTON1: //jezyk
            if(objMsg == BTN_MSG_PRESSED){ konfig_stan = KONFIG_JEZYK;
            BtnSetBitmap(pObj,(void*)&jezyk_sel8);Klik();}
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_CONFIG; }
            return 1;
    }
}

```



```

        case ID_BUTTON2:                //jasność
            if(objMsg == BTN_MSG_PRESSED){ konfig_stan = KONFIG_LAMPA;
            BtnSetBitmap(pObj,(void*)&jasnosc_sel8);Klik();}
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_CONFIG; }
            return 1;
        //case ID_BUTTON3:                //głośność dźwięków
        //    if(objMsg == BTN_MSG_PRESSED){ konfig_stan = KONFIG_NUTA;
        BtnSetBitmap(pObj,(void*)&jasnosc_sel8); Klik(); }
        //    if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_CONFIG; }
        //    return 1;
        //case ID_BUTTON5:                //info
        //if(objMsg == BTN_MSG_PRESSED){ BtnSetBitmap(pObj,(void*)&konfig_sel); konfig_stan =
        KONFIG_START; }
        //if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_INFO; }
        //    return 1;
        case ID_BUTTON6:                //wybierz PL
            if(objMsg == BTN_MSG_PRESSED){
                Klik();
                jezyk=POLSKI;
                ekran=CREATE_CONFIG;
            }
            return 1;
        case ID_BUTTON7:                //wybierz GB
            if(objMsg == BTN_MSG_PRESSED){
                Klik();
                jezyk=ANGIELSKI;
                ekran=CREATE_CONFIG;
            }
            return 1;

        case ID_BUTTON_T1:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =60;Klik(); }
            return 1;
        case ID_BUTTON_T2:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =240;Klik(); }
            return 1;
        case ID_BUTTON_T3:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =420;Klik(); }
            return 1;
        case ID_BUTTON_T4:
            if(objMsg == BTN_MSG_PRESSED){ czas_ozonowania_butli =599;Klik(); }
            return 1;

        case ID_BUTTON_MAIN:
            if(objMsg == BTN_MSG_PRESSED) { BtnSetBitmap(pObj,(void*)&powrot_szary8);
            konfig_stan = KONFIG_START;Klik(); }
            if(objMsg == BTN_MSG_RELEASED){ ekran=CREATE_MAIN; ZapisKonfiguracji();}
            //zapis konfiguracji po wyjściu z ekranu
            return 1;

        default:
            return 1;                                // default
    }
}
*/
//-----
void CreateEkranInfo(){

    GOLFree();

```



```

//PutImageExt(&ekr_ato_tlo, 0, 0);
SetCzasKoncaZabiegu(3);          // sek
OutTextXJustedY(0, 50, CENTRE_JUSTED, TextSel(jezyk,InfoProd), (void*)&Verdana_standard_12);
OutTextXJustedY(0, 90, CENTRE_JUSTED, TextSel(jezyk,InfoFlex), (void*)&Verdana_standard_12);
OutTextXJustedY(0,130, CENTRE_JUSTED, TextSel(jezyk,InfoKraj), (void*)&Verdana_standard_12);
testowy_display=0;
ZeroCzasZabiegu();
}
void ZmianyEkranuInfo(){
    if(!GetCzasKoncaZabiegu()){          //koniec wyświetlania
        ekran = CREATE_CONFIG;
        //if(flaga_testu){testowy_display=1;}
    }
    //if(GetCzasZabiegu()==20)PutImageExt(&ekr_czas_baza, 0, 0);      ;
    if(TupTouch()){
        ekran = CREATE_CONFIG;
        //if(flaga_testu){testowy_display=1;}
    }
}
//===== funkcje zabiegowe wspólne
=====
// Msg..., wspólne dla ekranów Pauzy i Erroru
WORD MsgEkranuPauzaError(WORD objMsg, OBJ_HEADER* pObj, STANY_EKRANU e_stop, STANY_EKRANU e_dalej){
    switch(GetObjID(pObj)){
        case ID_BUTTON1:          //stop
            //if(objMsg == BTN_MSG_PRESSED){      BtnSetBitmap(pObj,(void*)&stop_szark);    }
            if(objMsg == BTN_MSG_RELEASED){      ekran=e_stop;    }
            return 1;
        case ID_BUTTON2:          //dalej
            //if(objMsg == BTN_MSG_PRESSED){      BtnSetBitmap(pObj,(void*)&szary_downK);  }
            if(objMsg == BTN_MSG_RELEASED){      ekran=e_dalej;} //
    }
    czas_zabiegu=czas_pauza_zabiegu; }
    return 1;
    default:
        return 1;          // default
}
}

//=====
//===== STEROWANIA =====

void OutDaneZero(void){
    OFF_WY_1;
    OFF_WY_2;
    OFF_WY_3;
    OFF_WY_4;
    OFF_WY_5;
    OFF_WY_6;
    OFF_WY_7;
    OFF_WY_8;
}
void ZaworyOff(void){
    stan_V1=STAN_OFF;
    stan_V2=STAN_OFF;
    stan_V3=STAN_OFF;
    stan_V4=STAN_OFF;
    stan_V5=STAN_OFF;
    //stan_V6=STAN_OFF;
}

```

```

}

//===== USTAWIANIE STEROWAŃ =====
/*
    opis sygnałów magistrali wyjściowej
    sygnał      LE_1      LE_2,      LE_3
    -----
    xx_WY_1      ----,      -,      -
    xx_WY_2      stanO3,      -,      V5
    xx_WY_3      ---,      -,      -
    xx_WY_4      PK1,      -,      V4
    xx_WY_5      ----,      -,      V1
    xx_WY_6      ----,      -,      V2
    xx_WY_7      ----,      -,      -
    xx_WY_8      PK2,      -,      V3
*/
void IOdelay(void){
    ZerolOusek(); //zegar z taktem 10 mikrosekund
    while(GetlOusek()<=30); // 50 = 50 mikrosekund
}
void SterowanieSerwisowe(void){

    //sterowanie bezpośrednie wyjść, bez opóźnień dla włączania/wyłączania generacji ozonu i przekazników
    //==== wy1->St1 ....wy8->ST8 ====

    if(stan_PK2) {ON_WY_1;} else {OFF_WY_1;} //
    if(stan_PK1) {ON_WY_2;} else {OFF_WY_2;} //
    if(stan_O3) {ON_WY_3;} else {OFF_WY_3;} //
    if(stan_fan) {ON_WY_4;} else {OFF_WY_4;} //
    if(stan_pompa1) {ON_WY_5;} else {OFF_WY_5;} //
    if(stan_blue) {ON_WY_6;} else {OFF_WY_6;} //
    if(stan_green) {ON_WY_7;} else {OFF_WY_7;} //
    if(stan_red) {ON_WY_8;} else {OFF_WY_8;} //

    //IOdelay();
    LE_1_HIGH; // bufor-latch -st1, przepisanie magistrali na wyjścia bufora
    IOdelay(); // czas na ustawienie ze względu na EMI filtry 10 nF
    LE_1_LOW; // zapamiętanie.
    IOdelay(); // czas na separację/zamknięcie ze względu na EMI filtry 10 nF

    if(stan_V5) {ON_WY_1;} else {OFF_WY_1;} //
    if(stan_V4) {ON_WY_2;} else {OFF_WY_2;} //
    if(stan_V3) {ON_WY_3;} else {OFF_WY_3;} //
    if(stan_V2) {ON_WY_4;} else {OFF_WY_4;} //
    if(stan_V1) {ON_WY_5;} else {OFF_WY_5;} //
    {OFF_WY_6;} //
    {OFF_WY_7;} //
    {OFF_WY_8;} //

    //IOdelay();
    LE_3_HIGH; // bufor-latch st3, przepisanie magistrali na wyjścia bufora
    IOdelay();
    LE_3_LOW; // zapamiętanie.
    //IOdelay();
}

void SterowanieWy(void){
    //aktualizacja wyjść w pętli main
    //sterowanie pośrednie, z opóźnieniami dla włączania/wyłączania generacji ozonu i przekazników

```

```

//sterowanie ... - wyjścia bufora st1
//===== sterowanie przekaźnikami ===== zerujemy, a nast. ustawiamy odpowiednio gdy (stan_relayow)
{OFF_WY_1;}; //
{OFF_WY_2;}; //
if(stan_relayow) {
    if(stan_PK2) {ON_WY_1;}; //
    if(stan_PK1) {ON_WY_2;}; //
}
if(stan_generatora) {ON_WY_3;}; else {OFF_WY_3;}; //

//=====
if(stan_fan || stan_O3) {ON_WY_4;}; else {OFF_WY_4;}; // włączenie wentylatora
skojarzone ze włączeniem O3
if(stan_pompa1) {ON_WY_5;}; else {OFF_WY_5;};
if(stan_blue) {ON_WY_6;}; else {OFF_WY_6;}; //
if(stan_green) {ON_WY_7;}; else {OFF_WY_7;}; //
if(stan_red) {ON_WY_8;}; else {OFF_WY_8;}; //

LE_1_HIGH; // bufor-latch 1, przepisanie magistrali na wyjścia bufora
IOdelay();
LE_1_LOW; // zapamiętanie.
IOdelay(); // czas na separacje/zamknięcie ze względu na EMI filtry 10 nF

//sterowanie ... - wyjścia bufora st3
//ustawienie nowych stanów magistrali
if(stan_V5) {ON_WY_1;}; else {OFF_WY_1;}; //
if(stan_V4) {ON_WY_2;}; else {OFF_WY_2;}; //
if(stan_V3) {ON_WY_3;}; else {OFF_WY_3;}; //
if(stan_V2) {ON_WY_4;}; else {OFF_WY_4;}; //
if(stan_V1) {ON_WY_5;}; else {OFF_WY_5;}; //
{OFF_WY_6;}; //
{OFF_WY_7;}; //
{OFF_WY_8;}; //

LE_3_HIGH; // bufor-latch 3, przepisanie magistrali na wyjścia bufora
IOdelay();
LE_3_LOW; // zapamiętanie.
//IOdelay();
//fazowe włączenie i wyłączenie przekaźników stężenia i SSR-a zasilania
//gdy zmiana OFF->ON: włącz przekaźniki, odczekaj, włącz zasilania SSRa
//gdy zmiana ON->OFF: wyłącz SSRa, odczekaj 1/50sek. wyłącz przekaźniki

switch(faza_generacji){
    case OZON_OFF: //generator wyłączony
        stan_relayow=STAN_OFF; //PoziomOff();
        stan_generatora=STAN_OFF; //OFF_GENERATOR;
        if(stan_O3){ //właśnie ktoś włączył generację
            TmrOzonZero();
            faza_generacji=OZON_START;
        }
        break;
    case OZON_START: //czas włączania generatora
        stan_relayow=STAN_ON; //PoziomUstaw();
        stan_generatora=STAN_OFF; //OFF_GENERATOR;
        if(TmrOzon()>200){
            faza_generacji=OZON_ON;
        }
        break;
}

```

```

        case OZON_ON:                //generator włączony
            stan_relayow=STAN_ON;      //PoziomUstaw();
            stan_generatora=STAN_ON;    //ON_GENERATOR;
            if(!stan_O3){              //właśnie ktoś wyłączył generację
                TmrOzonZero();
                faza_generacji=OZON_STOP;
            }
        break;
        case OZON_STOP:              //czas wyłączenia generatora
            stan_relayow=STAN_ON;      //PoziomUstaw();
            stan_generatora=STAN_OFF;   //OFF_GENERATOR;
            if(TmrOzon()>200){
                faza_generacji=OZON_OFF;
            }
        break;
    }
}

void NadzorStanuButli(){
    switch(stan_butli){
        //case BUTLA_O3_BRAK: break;
        case BUTLA_O3_GRANICA:
            if(TmrButli())>=CZAS_BUTLA_O3BRAK){stan_butli = BUTLA_O3_BRAK; ZeroTmrButli(); }

            break;
        case BUTLA_O3_OK:
            if(TmrButli())>=CZAS_BUTLA_O3DOBRE){stan_butli = BUTLA_O3_GRANICA;}
            break;
    }
}

/*****
*****/

void OdczytStanuWejsc(){

#define filtr_ms 30                //czas - jednakowych odczytów wejścia - tu przycisku nożnego, filtracja drgań styków
// Poprzednie - zapamiętane, stany wejść
static char S4 = 1;
static char S2 = 1;

char stan;

    stan = IN_2;    //IN_2 = !nogal_yellow
    if(S2 != stan){
        TmrYellowZero();
        S2 = stan;
    }

    if(TmrYellow()>filtr_ms){
        nogal_yellow=!S2;
    }

    stan = IN_4;    //IN_4 = !nogal_blue
    if(S4 != stan){
        TmrBlueZero();
        S4 = stan;
    }

    if(TmrBlue()>filtr_ms){

```

```

        nogal_blue=!S4;
    }
}
//===== TRYB SERWISOWY -> procedury =====
//=====

void UstawJasnosc(WORD jasnosc){
    OC1RS = jasnosc*100;
    //OC1RS = tabela_jasnosci[jasnosc_led];
    //JASNOSC_LED_MIN do JASNOSC_LED_MAX
    //predefiniowane nieliniowo poziomy jasności zakres 0-ciemno , do 4000-pełna
}
void ZapisKonfiguracji(void){
    //zapis konfiguracji
    konfiguracja_aparatu[JASNOSC_LED] = jasnosc_led;
    konfiguracja_aparatu[WYBRANY_JEZYK] = jezyk;
    //konfiguracja_aparatu[VOLUME_DZWIEK] = volume_dzwiek;
    konfiguracja_aparatu[USER_CZAS] = czas_zabiegu;
    konfiguracja_aparatu[USER_POZIOM] = stezenie_ozonu;
    while(AT45_BUSY);
    main_write_through_buffer1(0,konfiguracja_aparatu,20);
    while(AT45_BUSY);
}
void OdczytKonfiguracji(void){
    //zapis konfiguracji
    while(AT45_BUSY);
    main_array_read(0,konfiguracja_aparatu,20);
    while(AT45_BUSY);
    //weryfikacja poprawności i podstawienie
    if ((konfiguracja_aparatu[JASNOSC_LED] < 1) || (konfiguracja_aparatu[JASNOSC_LED] > 40 ))
    {konfiguracja_aparatu[JASNOSC_LED]=30;}
    jasnosc_led = konfiguracja_aparatu[JASNOSC_LED];
    //---
    if ((konfiguracja_aparatu[WYBRANY_JEZYK] < POLSKI) || (konfiguracja_aparatu[WYBRANY_JEZYK] > ANGIELSKI ))
    {konfiguracja_aparatu[WYBRANY_JEZYK]=ANGIELSKI;}
    jezyk = konfiguracja_aparatu[WYBRANY_JEZYK];
    //---
    //if ((konfiguracja_aparatu[VOLUME_DZWIEK] < 1) || (konfiguracja_aparatu[VOLUME_DZWIEK] > 40 ))
    {konfiguracja_aparatu[VOLUME_DZWIEK]=23;}
    //volume_dzwiek = konfiguracja_aparatu[VOLUME_DZWIEK];
    //--- ostatnio ustawiny czas zabiegu 12-90 sekund
    if ((konfiguracja_aparatu[USER_CZAS] < CZAS_ZABIEGU_MIN) || (konfiguracja_aparatu[USER_CZAS] >
CZAS_ZABIEGU_MAX )) {konfiguracja_aparatu[USER_CZAS]=CZAS_ZABIEGU_SET;}
    czas_zabiegu = konfiguracja_aparatu[USER_CZAS];
    //--- ostatnio ustawine stezenia : 1,2,3.
    if ((konfiguracja_aparatu[USER_POZIOM] < 1) || (konfiguracja_aparatu[USER_POZIOM] > 3 ))
    {konfiguracja_aparatu[USER_POZIOM]=2;}
    stezenie_ozonu = konfiguracja_aparatu[USER_POZIOM];
}

void ImportDaty(char typ){
    //typ=1 bitmapy, typ = 2 dzwieki
    char temp,kolor;
    char bitmapa_zapis[1024];
    uint16_t m=0,s=0;
    uint16_t adres,reszta;
    uint32_t licznik=0; //ma być to co potem wyjdzie, narazie 0

    SetColor(BLUE);

```

```

ClearDevice();
kolor=1;
SetColor(RED);
SetFont((void*)&GOLFontDefault);

if (!(status_read())&(0x01)) {
    przestaw_na_512 ();          //PO PIERWSZYM WYWOŁANIU KONIECZNY RESET ZASILANIA
    SetColor(RED);
    SetFont((void*)&GOLSmallFont);
    OutTextXY(70,15,"Pamiec przestawiona.");
    OutTextXY(70,30,"Musisz na chwile wytlaczyc zasilanie.");
    while(1){};
}

if(typ==1){
    //pierwszy zapis konfiguracji
    SetColor(BLACK);
    while(AT45_BUSY);
    OutTextXY(70,5,"Konfig");
    main_write_through_buffer1(0,konfiguracja_aparatu,20);
    OutTextXY(170,5,"OK!");
    while(AT45_BUSY);
    //-----
    OutTextXY(70,25,"Import bitmap");
    adres=4; //startowy adres, omijam strone zerowq - danych konfiguracyjnych
    licznik =0;
}
if(typ==2){
    while(AT45_BUSY);
    //-----
    SetColor(WHITE);
    OutTextXY(70,25,"Import dzwiekow");
    adres=16000; //startowy adres komunikatow, dostepne 8192 stron, ustawiam polowe = 4000 stron *
    licznik =0;
}
RS232_init();
temp='J';
UART1WriteChar('Z');
while (UARTTimeoutChar(400000) != 'T') {
    UART1WriteChar('Z');
}
UART1WriteChar('O');
s=0;
while (s==0) {
    while (m<1024) {
        bitmapa_zapis[m]=UARTWaitChar();
        m++;
    }
    m=0;
    main_write_through_buffer1(adres,bitmapa_zapis,1024);
    while(AT45_BUSY); //czekaj na zakonczenie wewnetrznej operacji w pamieci
    adres=adres+4;

    SetColor(WHITE); //czyszczenie
    Bar(120,120,200,150); //czyszczenie pod licznik
    if (kolor) {kolor =0 ; SetColor(GREEN);}
    else {kolor =1 ; SetColor(RED); }
    OutTextXY(70,75,"Kolejna porcja");
}

```

```

        LiczbaToText((licznik++),7,&text_bufor[0]);
        OutTextXY(120,120,&text_bufor[0]);

        UART1WriteChar('N');

        if (UARTWaitChar()==0x00) {
            s=1;
        }
    }
    OutTextXY(120,170,"SUKCES");
    //-----
    ZeroMojDelay();
    while(GetMojDelay()<300){
        //opóźnienie dla odczytania końca
    }
}
//*****
//===== testowe - czasowe =====
void KalibracjaToucha(void){
    WORD x,y,r;
    r=10;
    TouchCalibration();
    SetFont((void*)&GOLMediumFont);
    SetColor(BLUE);
    ClearDevice();
    SetColor(WHITE);
    LiczbaToText((_calXMin),4,&text_bufor[0]);
    OutTextXY(50,20,&text_bufor[0]);
    LiczbaToText((_calXMax),4,&text_bufor[0]);
    OutTextXY(90,20,&text_bufor[0]);
    Circle(160,20,r);
    Line(160-5,20-5,160,20+8);
    Line(160+5,20-5,160,20+8);
    Circle(160,220,r);
    Line(160-6,220,160+6,220);
    LiczbaToText((_calYMin),4,&text_bufor[0]);
    OutTextXY(190,20,&text_bufor[0]);
    LiczbaToText((_calYMax),4,&text_bufor[0]);
    OutTextXY(240,20,&text_bufor[0]);
    while(1){
        x=TouchGetX();
        y=TouchGetY();
        if((x>3)&&(y>3)&&(x<316)&&(y<236))FillCircle(x,y,3);
        if((x>160-r)&&(x<160+r)&&(y>20-r)&&(y<20+r)) break;
        if((x>160-r)&&(x<160+r)&&(y>220-r)&&(y<220+r)){
            SetColor(BLUE);
            ClearDevice();
            SetColor(WHITE);
            Circle(160,20,r);
            Line(160-5,20-5,160,20+8);
            Line(160+5,20-5,160,20+8);
            Circle(160,220,r);
            Line(160-6,220,160+6,220);
        }
    }
}
//===== testowe - end =====

```


DOKUMENTACJA PROJEKTOWA - INFORMACJE POUFNE