1 Wstęp

Co to jest?
Hall2007 to oprogramowanie kontrolno sterujące dla komputerów PC. Hall komunikuje się ze specjalnie dla niego stworzonymi procesorami HallChip realizującymi funkcje wejść i wyjść cyfrowych i analogowych. Program może obsłużyć jeden procesor podłączony poprzez port RS232 lub sieć procesorów podłączonych pośrednictwem konwertera rs232/422.
Oprogramowanie pozwala na tworzenie aplikacji sterujących i pomiarowych oraz wizualizację i rejestrację ich pracy. Poza tworzeniem algorytmów sterujących (język pascal) i okien pełniących rolę paneli operatorskich i ekranów wizualizacji możemy skorzystać z wielu dodatkowych możliwości takich jak programatory czasowe, harmonogram zadań, rejestratory trendów i danych pomiarowych. Możemy sterować systemem za pomocą emaili, wysyłać emaile, tworzyć dynamicznie dokumenty HTML, sterować Winampem. Kilka programów może komunikować się ze sobą za pośrednictwem sieci LAN
Używając terminologii z dziedziny automatyki przemysłowej możemy powiedzieć że HALL2007 jest połączeniem sterownika Soft-PLC z systemem SCADA / HMI.

Hall2007 jest następcą Hall2002 opisanego w Elektronice Praktycznej 4/2004

1.1 Licencja programów systemu Hall2007

Oprogramowanie HALL2007 jest bezpłatne ale jego zastosowanie praktycznie nie ma sensu bez podłączenia do niego procesorów HallChip. Kupując procesor HALLCHIP nie płacisz za procesor - kupujesz LICENCJE na wykorzystanie oprogramowania HALL2007 !

2 Dokumentacja procesorów

Dokumentację procesorów wydzielono do oddzielnych dokumentów. Główny dokument Hall2007_procesory_HallChip.pdf opisuje wszystkie zagadnienia wspólnych dla wszystkich procesorów HalChip takich jak ich konfiguracja, podłączenie do portu komunikacyjnego, adresowania, wspólne dla wszystkich peryferia.

Konkretne procesory opisane są w szczegółowych dokumentach które należy traktować łącznie z dokumentem Hall2007_procesory_HallChip.pdf

3 Co potrafi HALL2007

3.1 Obsługa sieci procesorów

Program może obsłużyć sieć ośmiu procesorów HallChip. Można odłączyć jeden procesor do portu rs232, kilka procesorów (modułów sterujących) znajdujących się w niewielkiej odległości od siebie bez użycia dodatkowych konwerterów albo sieć modułów oddalonych od siebie o setki metrów przy zastosowaniu konwerterów rs232/rs422. Procesory podłączyć można za pośrednictwem konwertera rs232/USB lub konwertera rs232/LAN.
Istnieje szereg procesorów HallChip: Podstawowy procesory to HCHC1 który posiada 8wejść, 8wyjść 2 cyfrowe czujniki temperatury 3 wejścia analogowe, wejście szybkiego licznika, wejście RC5 i 2 wyjscia PWM. Dostępne są też inne procesory.

3.2 Realizacja algorytmów sterowania

Dzięki możliwości tworzenia skryptów sterujących w języku Pascal program może realizować wielorakie funkcje sterujące: od prostych zadań sterowniczych takich jak nadzór nad oświetleniem po skomplikowane algorytmy np. sterowanie automatycznymi testerami a nawet niektórymi maszynami. Niektóre z procesorów HallChip posiadają wbudowany prosty sterownik PLC co pozwala na pewną autonomie sterownika –np. procesor steruje bramą wjazdową a komputer nadzoruje tę bramę obserwując ją, rejestrując zamknięcia i otwarcia i sterując np. kilkoma bramami jednocześnie.

3.3 Wizualizacja, ekrany synoptyczne

Dzięki możliwości tworzenia ekranów z wykorzystaniem wielu komponentów takich jak mierniki, lampki, potencjometry, przełączniki i animowane grafiki możemy stworzyć dowolną wizualizację stanu sterowanych czy kontrolowanych obiektów. Możemy obejrzeć stan stref alarmu, zrobić tablicę z miernikami, wygodny panel do sterowania oświetleniem czy tablicę synoptyczną dla instalacji przemysłowej.

3.4 Pomiar i rejestracja wartości analogowych, wyjścia PWM

Podstawowe procesory HallChip mogą obsłużyć dwa cyfrowe czujniki temperatury DS18c20 i trzy wejścia analogowe. Program posiada zestawy filtrów i konwerterów do przetwarzania i przeliczania tych wartości oraz rejestratory trendów i danych pomiarowych do ich zapisu, prezentacji i przetwarzania. Dane można prezentować w układzie tabelarycznym, jako wykresy a także eksportować do plików excela.
Wyjścia PWM pozwalają na budowę przetworników cyfrowo – analogowych, sterować obrotami silników prądu stałego albo sterować jasnością oświetlenia LED.

3.5 Rejestracja zdarzeń

Rejestr zwany dziennikiem pozwala na rejestrację zdarzeń wraz z ich opisem, datą i godziną. Możemy zarejestrować czas załączenia jakiegoś urządzenia, moment otwarcia drzwi, uzbrojenia alarmu etc.

3.6 Programatory czasowe

Cztery proste i wygodne w obsłudze programatory dobowe i cztery programatory tygodniowe pozwalające na sterowanie w funkcji czasu – załączanie ogrzewania, oświetlenia, sterowanie nawadnianiem. Cztery timery pozwalają na jednorazowe ustawienie czasu działania (a zasadzie minutnika do jajek) lub uruchomienie jakiejś procedury za jakiś czas. Jest też „budzik" pozwalający np. na uruchomienie czajnika, światła i puszczenie muzyki z rana na pobudkę.

3.7 Harmonogram

Programatory czasowe pozwalają na stworzenie sekwencji cyklicznie wykonywanych czynności natomiast harmonogram pozwala na zaplanowanie ich listy – można określić ze dana czynność ma zostać wykonana o takiej to a takiej godzinie takiego to a takiego dnia. Stosując Halla do nadzoru nad domem możemy np. zaplanować sekwencję sterowania oświetleniem na czas wyjazdu albo stworzyć „indywidualny" program dla zraszaczy.

3.8 Poczta elektroniczna

Program może w reakcji na jakieś zdarzenia wysyłać e-maie, zarówno krótkie wiadomości mieszczące się w nagłówku listu jak i raporty tworzone na podstawie specjalnych matryc. Można system zaprogramować tak aby co godzinę wysyłał nam list z rozkładem temperatur w naszej szklarni.
Program można również sterować za pomocą e-maili. Wysyłamy na obserwowany przez program adres list w nagłówku którego umieszczamy hasło i numer lub symbol polecenia a program zrealizuje to polecenie.

3.9 Dynamiczne dokumenty HTML

Aby opublikować wyniki pomiarów w internetcie trzeba zapisać je do bazy sql, zainstalować serwer, napisać odpowiednie skrypty PHP.
Hall robi to trochę inaczej – tworzymy normalny kod strony www (dokument HTML) umieszczając w tekście specjalne znaczniki. Program kopiuje ten dokument z jednego katalogu do drugiego i zamienia znaczniki na aktualne wartości zmiennych tekstowych. Docelowy katalog może leżeć w obrębie naszego serwera ftp albo możemy za pomocą dołączonego klienta ftp wysłać na naszą stronę www. Dodatkowo program tworzy pliki jpg z obrazem wykresów oraz na rozkaz zapisać jako jpg wybrany obrazek z tablicy grafiki. Dzięki temu możemy uzupełnić nasz dokument HTML o dynamicznie zmieniane grafiki. Program posiada własną przeglądarkę WWW sterowaną rozkazami skryptów można więc np. wyświetlać dokumenty HTML lub strony WWW zależnie od stanu wejść czy potrzeb danego algorytmu.

3.10 Dźwięk, muzyka i mowa

Program może za pomocą instrukcji stosowanych w skryptach odtwarzać pliki wav, uruchamiać dowolne pliki audio lub video w systemowych odtwarzaczach oraz sterować odtwarzaczem Winamp. Może tez sterować głośnością i wyciszeniem dźwięku. Można np. stworzyć system sterowania nagłośnieniem podłączając wyjście audio karty muzycznej do odpowiednich wzmacniaczy.
Hall może współpracować z syntezatorem mowy Expressivo. jeżeli na komputerze jest zainstalowany i uruchomiony program Expressivo to można za pomocą odpowiedniej komendy „powiedzieć" dowolny tekst, na przykład mówiony raport o stanie urządzeń lub jaka jest temperatura na zewnątrz.

3.11 Praca w sieci LAN

Programy Hall mogą współpracować między sobą wysyłając sobie dane i polecenia za pośrednictwem sieci TCP/IP. Możemy zrobić tak że komputer odpowiedzialny za sterowanie funkcjami naszego domu za pomocą sieci modułów stał będzie w piwnicy a sterowanie i kontrolę będziemy realizowali za pomocą aplikacji które nie muszą pracować non stop a tylko wtedy gdy chcemy coś zrobić lub zobaczyć.

3.12 Sterowanie wyświetlaczami alfanumerycznymi

Jeden z procesorów HallChip pozwala na podłączenie jdno lub dwu wierszowych wyświetlaczy LCD do których program sterujący może wysłać celem wyświetlenia dowolny tekst. Teoretycznie można podłączyć 8 takich wyświetlaczy.

4 Instalacja i części składowe programu

4.1 instalacja

Program instalujemy za pomocą programu instalacyjnego HallSetup.exe. Podczas instalacji wybieramy katalog gdzie ma zostać zainstalowany program – domyślnie jest to typowy dla wszystkich programów katalog Program Files.

4.2 instalacja minimalna

Program skonstruowano tak że wszystkie potrzebne pliki i biblioteki posiada on w obrębie własnego głównego katalogu HALL2007 w którym został zainstalowany. Dlatego możliwa jest jego reinstalacja poprzez normalne skopiowanie całości plików i folderów w inne miejsce.
Normalnie po instalacji w folderach są wszystkie składniki które potrzebujemy do tworzenia i działania aplikacji. Jeśli jednak chcemy zainstalować program na komputerze docelowym gdzie instalacja ma po prostu działać bez możliwości jej edycji to niektóre składniki możemy pominąć.
Poniżej podano składniki katalogi (foldery) programu zaznaczając przez (-) te których nie musimy przenosić do docelowej aplikacji na innym komputerze (potrzebne są tylko do tworzenia aplikacji)

4.3 pliki i katalogi

- Folder BAZA – w folderze tym znajduje się plik bazy danych SQL z wynikami pomiarów, zapisami dziennika etc.
- Folder BIN – pliki pomocnicze dla menadżera projektów (-)
- Folder DOC – katalog z dokumentacją systemu (-)
- Folder HTMLSET – folder w którym umieszczamy matryce dokumentów html
- Folder HTMLOUT – folder w którym program umieszcza wygenerowane pliki html
- Folder INITL i folder UDF – foldery z elementami bazy danych
- Folder LIB – miejsce składowania bibliotek dla skryptu strującego
- Folder MAILER – folder z programami do obsługi poczty email
- Folder PLCEDIT – folder z programem narzędziowym i kodami programów wykonywalnych dla procesora HCHPLC (-)
- Folder PROJEKTY – w folderze tym znajdują się projekty aplikacji.
- Folder SET – folder danych aplikacji – zapis trendów, pamięci podręcznej, matryc e-maili etc.
- Folder SCRIPTINFO – folder z przykładami skryptów pascalowych
- Folder WAV – folder z plikami dźwiękowymi
- Plik Hall2007.exe – główny plik programu
- Plik HallProjectMenager.exe – plik programu zarządzającego projektami (-)
- Plik Projekt.ini – konfiguracja projektu (plik projektu aplikacji, numer portu rs, foldery dla htmla etc.)
- Plik Hall_tcp.ini – konfiguracja komunikacji tcp/ip
- Pliki *.DLL – biblioteki programu

5 Projektowanie i uruchomienie aplikacji

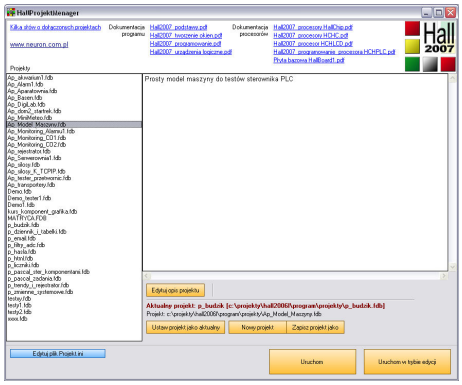
System składa się z dwu podstawowych programów – program hall2007.exe odpowiedzialny jest za tworzenie i „pracę” aplikacji a program HallProjectMenadżer.exe wspomaga zarządzanie projektami.

5.1 Projekt aplikacji

Program Hall2007 uruchomiony może zostać w dwu trybach – pracy i edycji.
Program uruchomiony bezpośrednio odczytuje parametr plik z pliku projekt.ini. Parametr ten określa nazwę projektu aplikacji która ma zostać załadowana i uruchomiona.

Program może zostać też uruchomiony z menadżera w trybie edycji w którym to możemy tworzyć i modyfikować projekty aplikacji

5.2 Menadżer projektów



Program menadżera pozwala na zarządzanie projektami znajdującymi się w podkatalogu Aplikacje, na tworzenie nowych aplikacji i zapis istniejących pod inną nazwą (jako nowy projekt).
Wskazany w menadżerze projekt możemy ustawić jako projekt aktualny – oznacza to że nazwa projektu zostanie wpisana do pliku projekt.ini i zostanie załadowana w momencie uruchomienia programu głównego.

Menadżer pozwala też na uruchomienie projektu w trybie pracy i w trybie projektowania. Jeżeli podczas uruchomienia wybrano z listy inny projekt niż aktualny projekt to zostanie on automatycznie uczyniony projektem aktualnym.

Z lewej strony widoczna jest lista projektów a z prawej strony opis zaznaczonego projektu.

W górnej części mamy linki do plików pdf z dostępną dokumentacją.

5.3 *Plik kprojekt.ini*

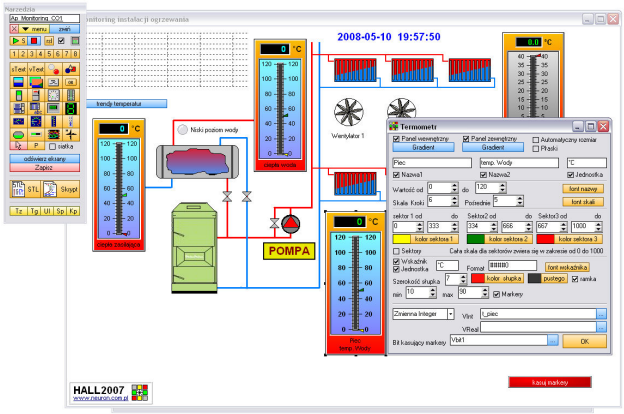
O tym który projekt aplikacji zostanie załadowany po uruchomieniu programu decyduje plik projekt.ini :

[PROJEKT]
plik=Ap_Monitoring_CO1 — Nazwa pliku projektu aplikacji (bez rozszerzenia) znajdującego się w podkatalogu projekty który będzie załadowany po uruchomieniu programu

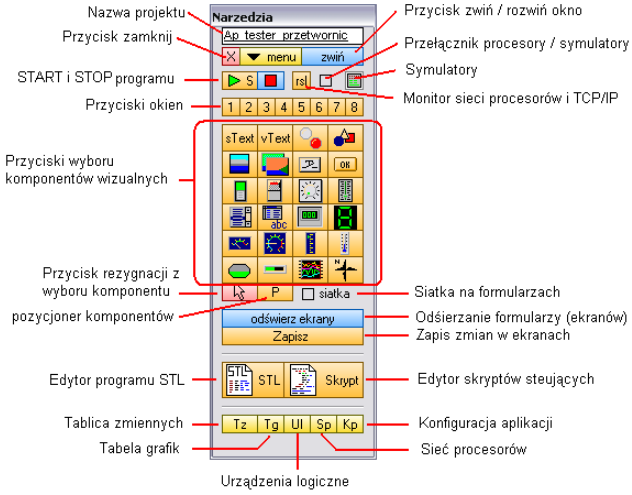
[PORTRS]
com=1 — Numer portu COM do którego podłączono procesor / sieć procesorów

[SYMULATOR]
symulatory=0 — Parametr określa czy pogram ma pracować w trybie komunikacji z procesorami (symulatory =0) czy w trybie pracy z symulatorami (symulatory =1)

5.4 *Tryb projektowania*



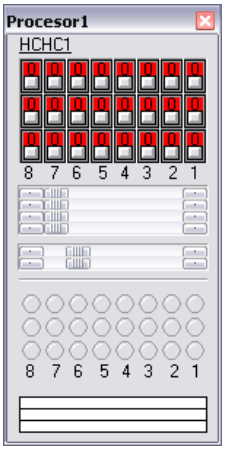
W trybie projektowania widoczne jest okienko narzędzi które pozwala na zarządzaniem projektem, jego edycję, edycję programów etc.



5.5 *Tryb pracy podczas projektowania*

Już podczas projektowania aplikacji działają niektóre jego funkcjonalności. Natomiast jeśli chcemy przetestować aplikację to możemy ją uruchomić za pomocą przycisku start – zostanie wtedy ograniczona wielkość okna narzędzi i otworzony port komunikacyjny lub otwarte okna z symulatorami procesorów.

6 Symulatory procesorów



Podczas konfiguracji procesorów możemy wybrać jeden lub dwa procesory do symulacji. Jeśli tak się stanie to o uruchomieniu aplikacji z załączonymi symulatorami pojawi się jedno lub dwa okna symulatorów.

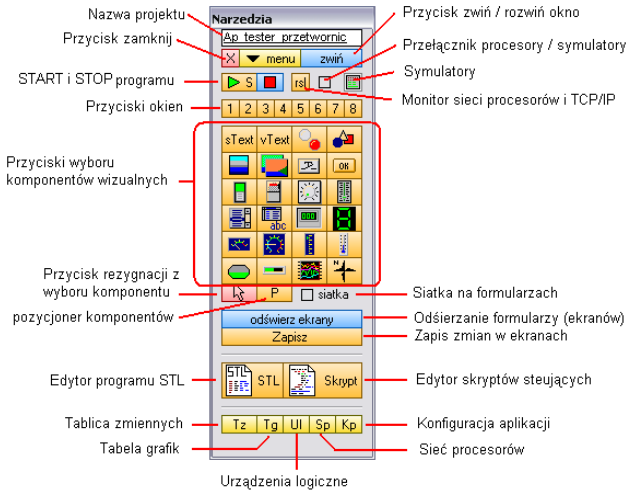
W tytule okna podany jest numer symulowanego procesora a poniżej jego typ.
Następnie mamy 24 przełączniki odpowiadające zmiennym in1 .. in24 procesora – ich przełączenie symuluje zmiany zmiennych przysyłanych przez procesor.
Następnie mamy 4 potencjometry odpowiadające zmiennym Ain1 .. Ain4 i dwa potencjometry odpowiadające zmiennym r1 i r2

Poniżej mamy 24 ledy odpowiadające zmiennym out1 ..out24 i trzy bargrafy odpowiadające zmiennym Aout1, 2 i 3.

Po postawieniu kursora nad każdym z elementów wyświetlany jest „dymek" z opisem zmiennej i jej funkcji do której jest dany element przyporządkowany.

W trybie edycji symulatory możemy przywołać przyciskiem na okienku narzędziowym natomiast w trybie pracy pojawia się napis symulator na logo programu w dolnym lewym rogu pierwszego okna.

7 Okienko Narzędzia



W trybie projektowania widoczne jest okienko z narzędziami.
Okienko można zwinąć aby nie przeszkadzało w pracy.
Znajdują się na nim przyciski sterujące aplikacją, przyciski wywołujące okna z edytorami programu, okna z konfiguracją zmiennych, urządzeń logicznych, procesorów.

Okno narzędzi stanowi też zasobnik z komponentami do budowy ekranów.
Jeśli chcemy na projektowanym oknie postawić jakiś komponent to najpierw klikamy przycisk odpowiedniego komponentu a następnie w okno w miejscu gdzie komponent ma się pojawić.
Możemy wycofać się z postawienia komponentu za pomocą przycisku rezygnacji.

8 Okna (ekrany)

8.1 Wybór okna w trybie projektowania i wywołanie podczas pracy

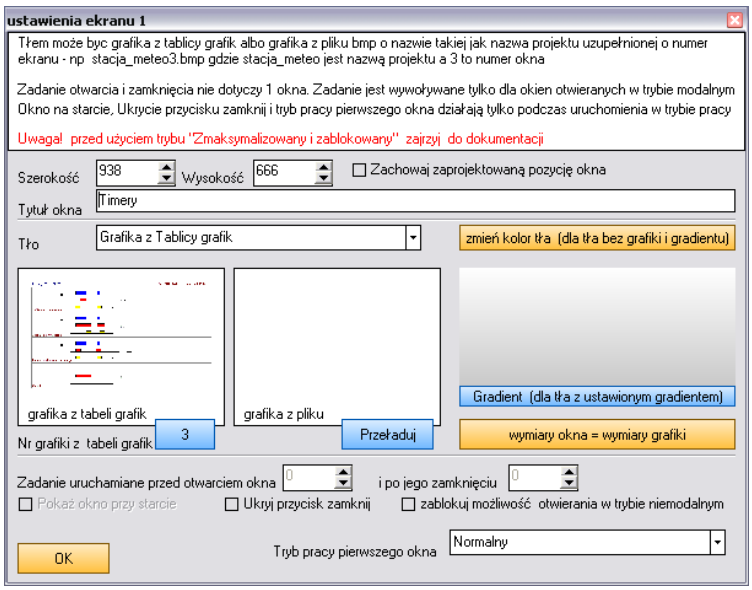
Po uruchomieniu programu zawsze widoczne będzie okno nr 1 – będzie ono głównym oknem aplikacji a jego zamknięcie będzie jednoznaczne z zakończeniem pracy programu. Inne okna otwieramy za pomocą odpowiednio skonfigurowanego przycisku sterującego lub funkcyjnego.
W trybie projektowania widoczne jest okienko narzędziowe a na nim mamy osiem ponumerowanych przycisków do wywoływania odpowiednich okien.

8.2 Okno modalne i nie modalne

Okno projektu a także niektóre z okien urządzeń możemy otworzyć jako okno modalne albo nie modalne. Co to oznacza ?
Okno modalne to takie okno które podczas swego działania blokuje operacje w innych oknach programu. Tak długo jak okno modalne jest widoczne tak długo nie możemy uzyskać dostępu do innych okien programu pomimo iż je widzimy.

8.3 Parametry okien

Aby zmienić parametry danego okna kliknij w nie dwukrotnie – pojawi się wtedy formularz z ustawieniami



Szerokość i wysokość okna – rozmiar okna możemy zmienić za pomocą myszy (jeszcze przed otworzeniem formularza parametrów) w ten sposób że chwytamy za prawą lub dolną krawędź okna i rozciągamy do pożądanych rozmiarów. Możemy też ustawić wysokość i szerokość okna za pomocą nastawników.

Położenie okna po uruchomieniu programu - jeśli załączymy opcję zachowaj zaprojektowaną pozycję okna to okno pojawi się tam gdzie było w momencie zapisu projektu (przycisk zapisz na oknie narzędziowym). W przeciwnym przypadku okno będzie ustawione na środku ekranu.

Tytuł okna jest wyświetlany w jego belce tytułowej.

Tło okna – są cztery opcje do dyspozycji:

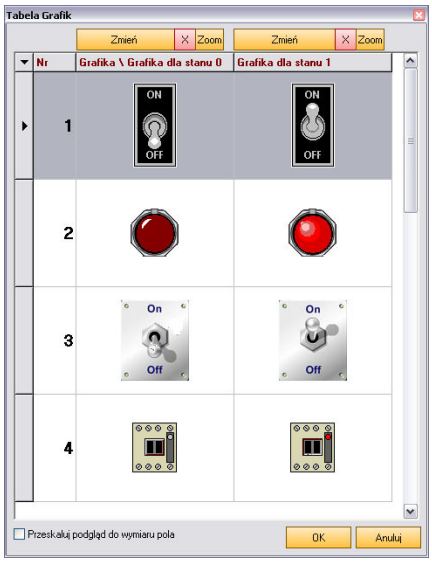
- Tło bez grafiki – wybieramy jedynie kolor tła
- Grafika z Tablicy grafik – na oknie zostanie wyświetlona grafika pobrana z tablicy grafik z pozycji o wybranym numerze
- Grafika z pliku "nazwaprojektu_nrekranu.bmp"
- Tło gradientowe

Trzecia opcja działa w ten sposób że w podkatalogu projekty poszukiwany jest plik BMP o takiej samej nazwie jak nazwa projektu i numerze okna.
Powiedzmy ze nasz projekt nazywa się idom – wtedy wszystkie informacje znajdują się w pliku idom.fdb. Jeżeli stworzymy obraz w formacie bitmapy z tłem i zapiszemy w katalogu projekty pod nazwą idmo5.bmp to będzie mógł on zostać użyty jako tło 5 okna tego projektu.
Tryb ten jest bardzo pożyteczny wtedy kiedy planujemy częste zmiany tła podczas projektowania – możemy wtedy w każdej chwili edytować plik tła a po zapisie zmian wprowadzić je przyciskiem przeładuj. Na koniec jak tło jest kompletne możemy je zapisać do tabeli grafik, zmienić ustawienie tła a plik usunąć
Na formularzu mamy podgląd zarówno wybranej grafiki z tabeli grafik jak i z pliku jeśli takowy został przygotowany.
Możemy też automatycznie dostosować rozmiar okna do rozmiaru grafiki tła ale musi ono mieć rozmiar co najmniej 100 na 100 pikseli

Każde otwarcie i zamknięcie okna w trybie modalnym może powodować wywołanie odpowiedniego zadania – wystarczy wpisać odpowiedni, różny od zera numer zadania.
Opcja pokaż okno przy starcie powoduje jego automatyczne wyświetlenie w trybie nie modalnym o ile tryb nie modalny nie zostanie zablokowany.
Blokada trybu modalnego uniemożliwia jego użycie np. w przycisku funkcyjnym wywołującym okno nawet gdyby taką opcję wybrano.
Parametr pokaz na starcie nie dotyczy pierwszego okna – zawsze jest ono widoczne po starcie programu.

Tryb pracy pierwszego okna – pierwsze okno projektu może mieć pewne szczególne właściwości :

9 Tablica grafik



Aby skorzystać w projekcie z grafiki (poza obrazem statycznym – zobacz komponent grafika) musimy załadować ją do tablicy grafik.
Tablica grafik może przechować 128 x 2 grafik. Razy dwa bo w jednym wierszu można przechowywać grafikę dla dwu stanów: 0 i 1.

Do tablicy można załadować dowolny plik graficzny w formacie BMP. Należy jedynie uważać aby nie przesadzić z wielkością pliku – nie zaleca się ładować do tablicy plików większych niż 2 Mb.

Aby załadować nową (lub zmienić istniejącą na inną) grafikę użyć należy przycisku zmień który otworzy okienko wyboru pliku. Każda z kolumn ma swój własny przycisk do zmiany grafiki nad tabelą a zmieniana jest grafika w tym wierszu który jest podświetlony.
Możemy też usunąć grafikę lub wyświetlić ją w oknie podglądu przyciskiem zoom.

Pod tabelką znajduje się przełącznik Przeskaluj podgląd – kiedy grafika jest większa niż komórka tablicy to widzieć będziemy tylko jej fragment – przełącznik ten pozwoli na takie przeskalowanie (podglądu a nie samej grafiki) aby cała grafika była widoczna w komórce.

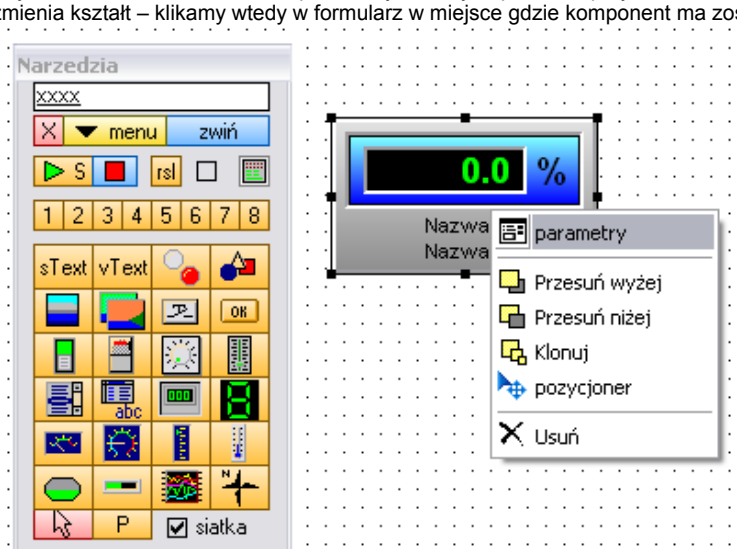
Tam gdzie używana jest tylko jedna grafika – p dla animacji albo jako tło okna – wpisujemy ją tylko do lewej kolumny.

10 Komponenty

Na formularzach umieszczamy komponenty wizualne które stanowić będą elementy ekranu – przyciski, przełączniki, kontrolki,grafiki itp.

10.1 Tworzenie komponentów, zmiana położenia i parametrów, przycisk rezygnacji

Aby na formularzu umieścić komponent wybieramy odpowiedni przycisk w okienku narzędziowym. Po kliknięciu w dany przycisk komponentu kursor zmienia kształt – klikamy wtedy w formularz w miejsce gdzie komponent ma zostać utworzony.



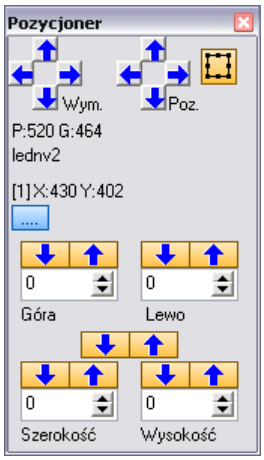
Aby przesunąć obiekt zaznaczamy go myszką i przesuwamy trzymając naciśnięty jej prawy przycisk. Aby dokładnie określić położenie komponentu można użyć klawiszy ze strzałkami które przesuwają go o jeden piksel.

Lewym przyciskiem myszy otwieramy menu które pozwala na

- Otwarcie okna z parametrami komponentu
- Przesuniecie wyżej komponentu przesłoniętego innym
- Przesunięcie komponentu nachodzącego na inny pod niego
- Sklonowanie (skopiowanie) komponentu – powstaje wtedy identyczny komponent który można przesunąć w inne miejsce i zmienić jego parametry
- Usunąć komponent
- Otworzyć okno parametrów okna (formularza)

Czerwony przycisk ze strzałką pozwala na zrezygnowanie z rozpoczętej już procedury tworzenia nowego komponentu

10.2 Pozycjoner



Pozycję komponentu na formularzu możemy ustalić za pomocą myszy, za pomocą przycisków kierunkowych z klawiatury i za pomocą pozycjonera.
Poza precyzyjną zmianą pozycji (przyciski ze strzałkami zmieniają pozycję o jeden piksel) możemy za pomocą pozycjonera zmienić wymiar komponentu.
Wyświetlane w okienku pozycjonera są: położenie komponentu (lewego górnego narożnika), numer okna oraz położenie myszy na oknie.

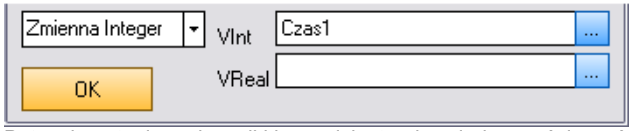
Po rozwinięciu drugiej części okienka pozycjonera przyciskiem [...] mamy do dyspozycji dwa narzędzia ułatwiające nam ułożenie komponentów na ekranie.
Pierwszy zestaw nastawników ze strzałkami działa w ten sposób że klikając strzałkę skierowaną do dołu wpisujemy (zapamiętujemy) aktualne położenie komponentu. Strzałką skierowaną do góry przypisujemy położenie do aktualnie wybranego komponentu. Możemy wiec zapamiętać np. położenie w pionie jednego komponentu a następnie wymusić to samo położenie dla innych komponentów.

Analogicznie możemy zapamiętać wymiary komponentów aby przypisać je do innych komponentów.

10.3 Blokowanie pozycji komponentów

Trzy komponenty: obrazek, figura i pole gradientowe mają właściwość ustalaną w oknie parametrów pozwalająca na zablokowanie ich położenia w trybie edycji. Na komponenty te będziemy często nakładali inne komponenty a co za tym idzie możemy niechcąco zmienić ich położenie co czasami bywa kłopotliwe. Dlatego możemy zablokować możliwość ich przesuwania za pomocą myszy. Nadal możemy zmieniać pozycję komponenty za pomocą klawiszy kierunkowych i pozycjonera.

10.4 Przypisywanie zmiennych do komponentów i aktywacja zadań z komponentów



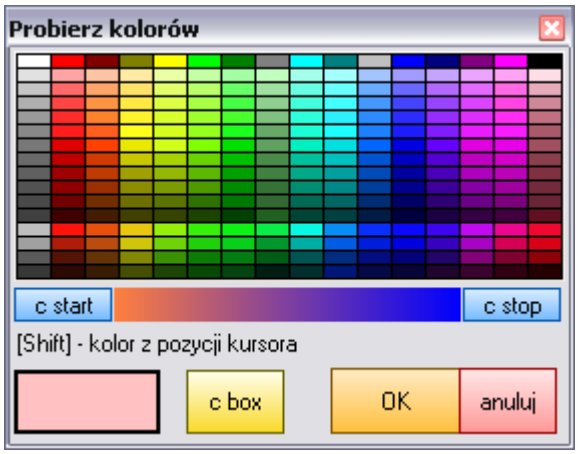
Komponenty (poza tekstem statycznym) zmieniają albo reprezentują stan wybranych zmiennych. Led świeci wtedy kiedy załączony jest skojarzony z nim bit, mierniki wyświetlają stan odpowiednich zmiennych.

Potencjometry i przełączniki i przycisk sterujący jednocześnie wyświetlają i sterują zmiennymi – jeżeli skojarzymy dwa przełączniki z tą samą zmienną to zmiana stanu jednego zmieni stan drugiego. Jeżeli „pokręcimy" potencjometrem to zmienimy wartość jakiejś zmiennej ale jeśli zmienną tę zmieni jakieś zadanie w skryptach to potencjometr sam się odpowiednio ustawi.

Aby wybrać zmienną dla komponentu to w oknie jego parametrów wyszukać musimy pole opisujące wybraną zmienną i kliknąć w niebieski przycisk na jego końcu – otwarta zostanie wtedy tabela zmiennych. Wybieramy wtedy tę odpowiednią.

Niektóre komponenty mają dwa pola edycyjne wyboru zmiennych – dla wartości typu Vint i Vreal oraz przełącznik który pozwala wybrać którą zmienną ma komponent obsługiwać
W trybie projektowania nad komponentami po najechaniu kursorem myszy wyświetlana jest jego nazwa i zmienna (zmienne) która jest z nimi skojarzona.
Niektóre komponenty, np. przyciski lub przełączniki po ich zmianie stanu uruchamiają zadanie o ustalonym numerze. Przykładowo dla przełącznika możemy ustawić nr zadania na 21 – wtedy po każdej zmianie stanu przełącznika wywołane zostanie zadanie Task21.
Jeżeli klikniemy podwójnie w pole zmiany numeru zadania a nie jest ono zerowe to nastąpi zamknięcie okna konfiguracji i przejście do edytora skryptu oraz odszukanie stosownego zadania.

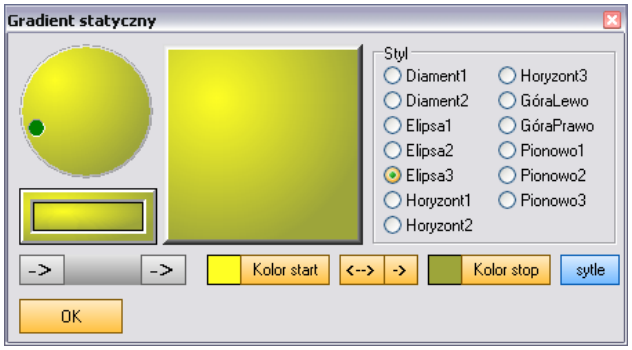
10.5 Okienko dialogowe zmiany kolorów



Prawie wszystkie obiekty mają ustawiany parametr, parametry kolor. Ponieważ często kolor ten chcemy zgrać z kolorem tła, albo na odwrót kolor tła z kolorem jakiegoś składnika obiektu to zamiast standardowego okienka wyboru kolorów zastosowano dialog wyboru koloru z probierzem.
Działa to tak, trzymając wciśnięty klawisz shift pokazujemy kursorem myszy dowolne miejsce na ekranie, a probierz odczytuje kolor wyświetlany na ekranie w miejscu położenia kursora.
Puszczając shift zapamiętujemy ostatnio wskazany kolor.

Możemy również po prostu kliknąć w odpowiedni kolor na palecie kolorów. Kolor możemy odczytać również z paska z gradientem. Przyciski start i stop ustalają bieżący kolor jako kolor zaczynający lub kończący pasek gradientu. Można też otworzyć standardowy dialog wyboru koloru przyciskiem c box

10.6 Gradienty

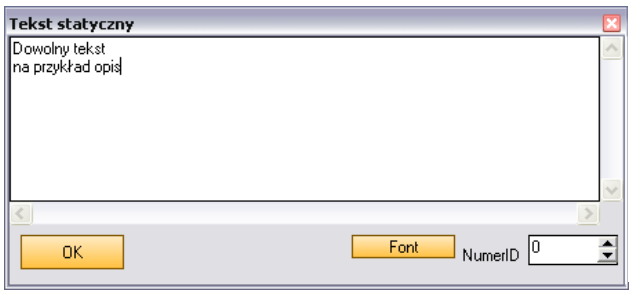


Wiele komponentów wizualnych bazuje na płaszczyznach pokrytych gradientem. Wszędzie tam gdzie w ustawieniach komponentu zdecydujemy się na zmianę gradientu pokaże się okno dialogowe z jego parametrami.

O wyglądzie gradientu decydują trzy parametry – kolor początkowy, kolor końcowy oraz styl. Kolory zmieniamy za pomocą dialogów koloru a styl wybieramy z listy styli. Możemy zamienić kolory miejscami przyciskiem [< -- >]
lub wpisać kolor startowy do koloru początkowego przyciskiem [->] aby uzyskać jednolitą powierzchnię gradientu.

Niebieski przycisk pozwala na otwarcie okna z gotowymi stylami gradientów
Pole z dwoma przyciskami [->] pozwala na zapamiętanie i późniejsze użycie wzoru gradientu

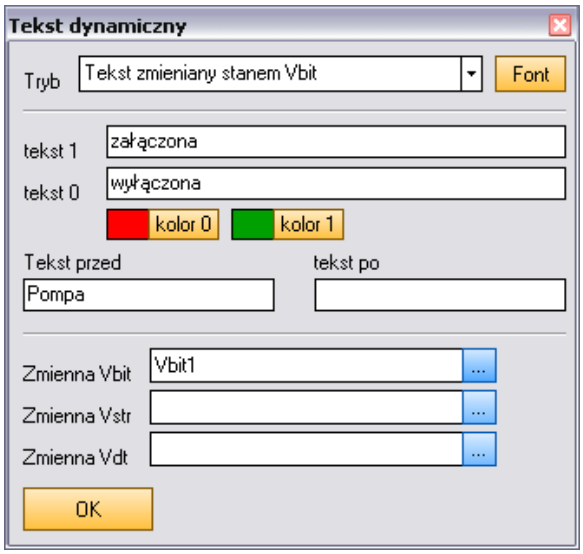
10.7 Komponent Tekst statyczny



Komponent tekst statyczny jak nazwa wskazuje jest zwykłym tekstem który możemy umieścić w dowolnym miejscu formularza.
Możemy zmienić czcionkę tekstu wraz z jej rozmiarem, kolorem i stylem.

Numer ID pozwala na manipulowanie tekstem i kolorem z poziomu programu – zobacz rozdział operowanie komponentami z poziomu programu.

10.8 Komponent Tekst dynamiczny

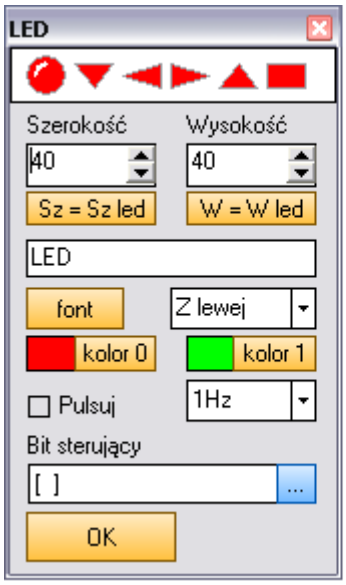


Komponent Tekst dynamiczny może pracować w kilku trybach :

- wyświetla wybraną zmienną tekstową Vstr
- wyświetla wybrany napis w wybranym kolorze zależny od stanu zmiennej bitowej Vbit
- Wyświetla w jednym z wybranych formatów wybraną zmienną czasu Vdt
- Wyświetla w jednym z wybranych formatów aktualny czas i/lub datę
- Wyświetla datę i czas uruchomienia programu
- Wyświetla nazwę aktualnej aplikacji

Poza wyborem odpowiedniej zmiennej możemy zmienić następujące parametry:
Wyświetlane teksty dla stanu 0 i 1 zmiennej Vbit
Stały tekst wyświetlany przed i po tekście ustalonym dynamicznie
Czcionkę
Kolor dla stanu 0 i 1 zmiennej Vbit

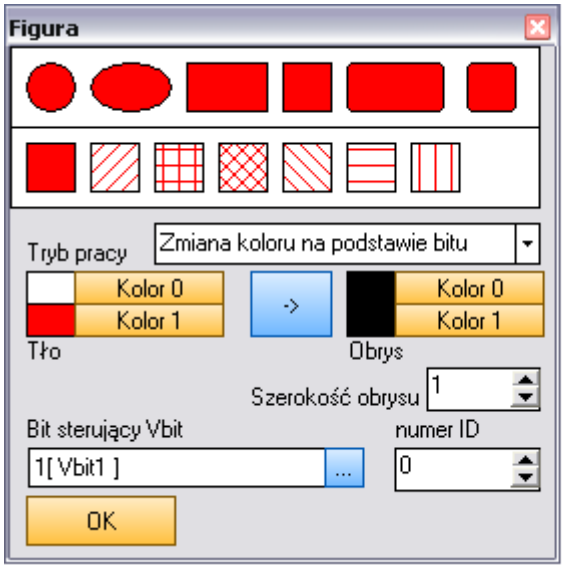
10.9 Komponent LED

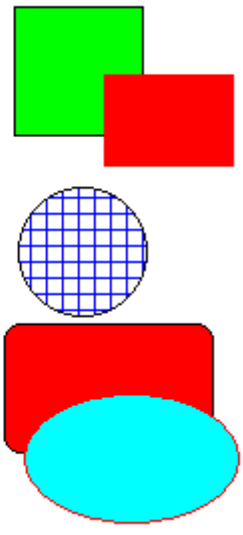


Funkcje komponentu tłumaczy w zasadzie sama nazwa – led świeci lub nie świeci zależnie od stanu skojarzonej z nim zmiennej bitowej Vbit.

Parametry komponentu:
Kształt – wybieramy klikając w wybrany led u góry okna
Szerokość i wysokość – dla kontrolki okrągłej tylko szerokość – określa rozmiar widocznej kontrolki. Rozmiar komponentu może być większy niż rozmiar kontrolki w celu pokazania umieszczonego obok napisu. Rozmiar komponentu ustalamy przez odpowiednie rozciąganie go myszą.
Przycisk Sz= Sz led ustala szerokość komponentu = szerokości kontrolki
Przycisk W = W led ustala wysokość komponentu = wysokość kontrolki
Napis wyświetlany obok kontrolki, jego czcionka i umiejscowienie względem kontrolki.
Kolor 0 i Kolor 1 – kolory kontrolki dla stanu o i 1 bitu sterującego
Pulsuj – załączenie pulsowania kontrolki (dla stanu 1) oraz tempo pulsowania.
Bit sterujący (zmienna Vbit) wybrany z tablicy zmiennych

10.10 Komponent Figura





Komponent figura zmienia kolor na podstawie stanu sterującego bitu.

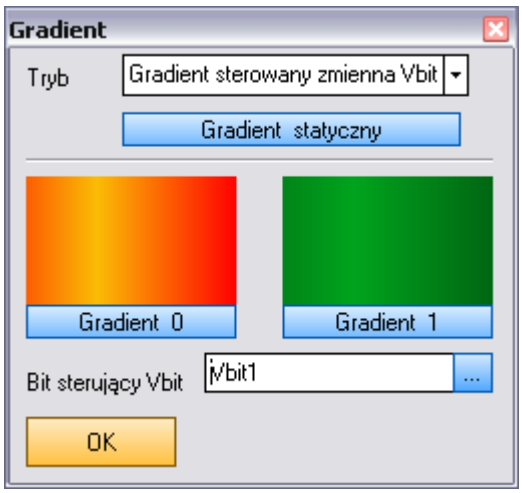
Parametry komponentu:
Kształt i wypełnienie – zmiana przez kliknięcie w odpowiednie pole
Tryb pracy:

- Zmiana koloru na podstawie stanu bitu
- Statyczny (może w tym stanie być sterowany programowo)

Kolor tła i kolor obrysu dla stanu 0 i 1 sterującego pitu
Bit sterujący
Szerokość obrysu
Numer ID – parametr dla rozkazów z skryptu

Kolor figury można sterować programowo za pomocą odpowiednich rozkazów co opisano przy okazji opisu języka programowania

10.11 Komponent Pole gradientowe

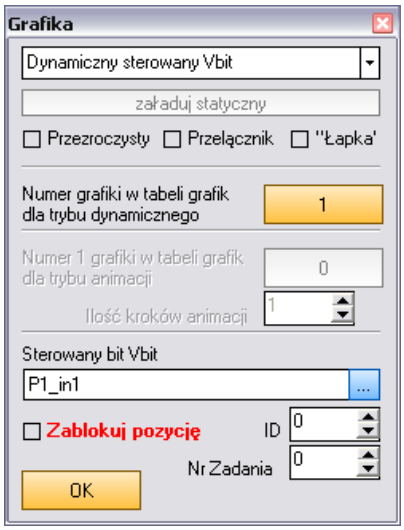


Komponent Gradient może pracować jako komponent pasywny i stanowić element tła lub grafiki albo jako aktywny sterowany stanem zmiennej bitowej Vbit.

W trybie pasywnym wybieramy styl gradientu za pomocą formularza gradientów otwieranego przyciskiem gradient statyczny.

Dla gradientu dynamicznego poza wyborem zmiennej sterującej ustalamy styl gradientu dla stanu 0 i 1 zmiennej sterującej.

10.12 Komponent Grafika



Komponent służy do wyświetlania

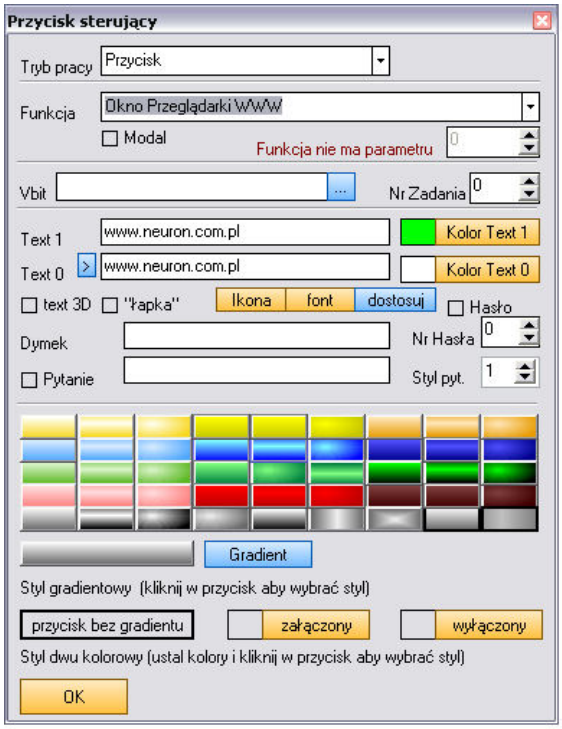
- grafiki statycznej – załadować można dowolny plik w formacie BMP lub JPG
- grafiki dynamicznej – wyświetlana jest grafika z tablicy grafik o podanym numerze – dla wartości 1 wskazanej zmiennej bitowej wyświetlana jest grafika z prawej a dla wartości 0 z lewej kolumny tabeli.
- grafiki animowanej – podajemy numer grafiki z tablicy grafik i ilość kroków animacji – jeśli wybrana zmienna sterująca będzie miała stan 1 to komponent będzie cyklicznie wyświetlał grafiki od podanego numeru do numeru + ilość kroków

Wyświetlana grafika może być przezroczysta – znaczy to że ten kolor który ma pixel w lewym dolnym rogu grafiki będzie przezroczysty. Jeżeli mamy grafikę na białym tle tak to wszystko co jest białe na tej grafice będzie przezroczyste.

Grafika może też pełnić rolę przełącznika – kliknięcie w obrazek spowoduje wtedy przełączenie na przeciwny stan sterującej zmiennej a co za tym idzie zmianę obrazka na inny.
Po kliknięciu w obrazek może zostać wywołane ustawione zadanie .

Możliwe jest też zablokowanie pozycji obrazka – nie da się go wtedy w trybie edycji przesunąć ani zmienić jego rozmiaru.
Numer ID jest numerem dla procedur sterujących obrazkiem w sposób bezpośredni.

10.13 Komponent Przycisk sterujący



Komponent przycisk może pracować w kilku trybach:

- Przycisk – naciśnięcie przycisku powoduje zmianę zmiennej bitowej tak długo jak przycisk jest naciskany
- Przełącznik – kolejne naciśnięcia zmieniają stan przycisku a co za tym idzie stan zmiennej
- Przycisk funkcyjny – nie jest sterowana żadna zmienna a wywoływana jest określona funkcja (zobacz niżej funkcje przycisków)
- Indykator – przycisk pełni tylko funkcję wskaźnika

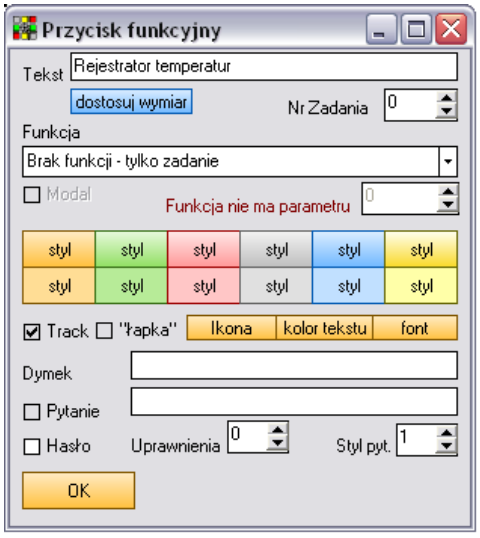
Jeżeli numer zadania jest różny od zera to każde naciśniecie przycisku powoduje wywołanie tego zadania.

Parametry przycisku:
Tekst i kolor czcionki tekstu dla wartości 0 i 1 zmiennej.
Tekst 3D – załączenie powoduje „zagłębienie" tekstu
Łapka – załączenie spowoduje że kursor nad przyciskiem przyjmie formę naciskającej dłoni

Dymek – treść podpowiedzi która ukaże się po najechaniu kursorem myszy nad przycisk
Pytanie – załączenie pytania i wpisanie jego treści spowoduje że po naciśnięciu przycisku zanim zostanie wykonana przypisana mu funkcja pojawi się okienko dialogowe z zapytaniem. Można też ustalić styl okienka z zapytaniem.
Hasło i numer hasła – jeśli hasło zostanie załączone to po naciśnięciu przycisku zostanie wyświetlone zapytanie o hasło. Aby funkcja została zrealizowana hasło musi być na liście haseł a jego posiadacz musi mieć poziom co najmniej równy lub większy jak numer hasła.

Przycisk dostosuj pozwala na automatyczne dostosowanie wymiaru przycisku do tekstu.
Styl i kolory przycisku możemy ustalić na dwa sposoby. Albo przez kliknięcie w jeden z przycisków z gradientem „przenosimy" jego styl na przycisk albo wybieramy przycisk bez gradientu. Jeśli wśród przygotowanych przycisków gradientowych nie ma interesującego nas stylu to poniżej mamy przycisk którego styl możemy ustalić sami. Możemy również ustalić kolory dla przycisku bez gradientu.

10.14 Komponent Przycisk funkcyjny



Przycisk funkcyjny pozwala na uruchomienie wybranej funkcji i / lub uruchomienie określonego zadania.

Parametry przycisku:
Tekst – wyświetlany tekst na przycisku
Funkcja i parametr funkcji – zobacz opis poniżej
Style przycisków – przenosimy je na edytowany przycisk przez kliknięcie w przycisk o pożądanym stylu.
Track – określa czy przycisk ma zmieniać kolor po najechaniu na niego myszą
Łapka – załączenie spowoduje że kursor nad przyciskiem przyjmie formę naciskającej dłoni

Dymek – treść podpowiedzi która ukaże się po najechaniu kursorem myszy nad przycisk
Pytanie – załączenie pytania i wpisanie jego treści spowoduje że po naciśnięciu przycisku zanim zostanie wykonana przypisana mu funkcja pojawi się okienko dialogowe z zapytaniem. Można też ustalić styl okienka z zapytaniem.
Hasło i numer hasła – jeśli hasło zostanie załączone to po naciśnięciu przycisku zostanie wyświetlone zapytanie o hasło. Aby funkcja została zrealizowana hasło musi być na liście haseł a jego posiadacz musi mieć poziom co najmniej równy lub większy jak numer hasła.

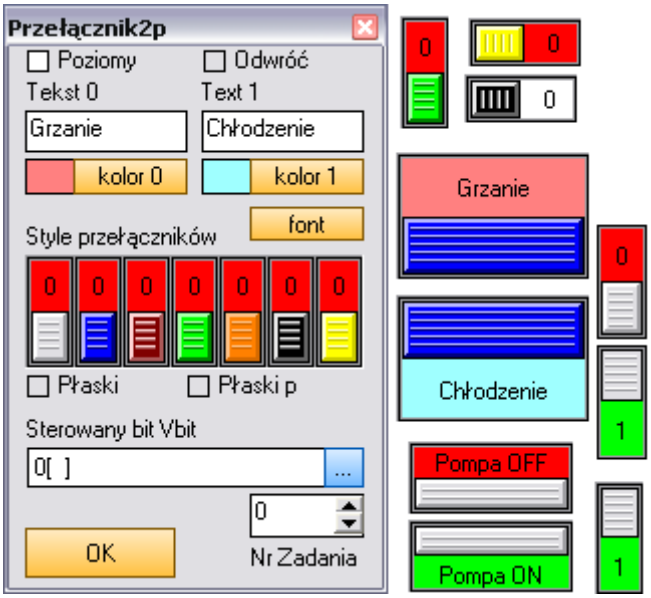
10.15 Funkcje przycisków

Przyciski sterujący i funkcyjny może wywołać (poza wykonaniem zadania ze skryptu) konkretne działanie:
Dla wybranej funkcji dostępne są dwa parametry: modal – współpracuje z niektórymi poleceniami i pozwala na wybór czy okno ma być modalne czy nie – głównie dla funkcji wywołania okien-formularzy. Drugi parametr to liczba której znaczenie jest zależne od konkretnej funkcji i opisane po jej zmianie

- Okno [nr] – Powoduje otwarcie okna o wskazanym numerze
- Zamknij okno – Zamyka okno na którym znajduje się przycisk. jeśli znajduje się na oknie numer 1 to zamknie program.
- Ukryj program do ikony – Ukrywa program, aby przywrócić jego widoczność trzeba kliknąć w ikonę w systemowym pasku narzędziowym
- Programator dobowy [nr] – Otwiera okienko wskazanego programatora dobowego
- Programator tygodniowy [nr] - Otwiera okienko wskazanego programatora tygodniowego
- Timer programowalny [nr] - Otwiera okienko wskazanego timera
- Okno Alarmów - Otwiera okienko alarmów
- Okno Dziennika [grupa] - Otwiera okno dziennika dla wskazanej grupy – grupa nr 4 to wszystkie grupy
- Okno Harmonogramu – Otwiera okno harmonogramu
- Okno Rejestratora [nr] - Otwiera okno rejestratora danych dla wskazanej grupy pomiarów
- Okno Rejestratora Rezerwa
- Okno Przeglądarki WWW – Otwiera okno wbudowanej przeglądarki www
- Okno Zegara - Budzika
- Hasła i uprawnienia – Otwiera okno edytora haseł i uprawnień
- Okno pomocy – otwiera okno z plikiem pomocy (plik rtf o takiej samej nazwie jak nazwa projektu znajdujący się w katalogu projektu)

- Monitor sieci procesorów – otwiera okno z monitorem transmisji i konfiguracji sieci procesorów oraz połączenia TCP/IP
- rezerwa
- Wprowadź liczbę INT do zmiennej Vint [nr] – otwiera okno wprowadzające liczbę całkowitą do zmiennej
- Wprowadź liczbę Real do zmiennej Vreal [nr] – otwiera okno wprowadzające liczbę rzeczywistą do zmiennej
- Wprowadź tekst do zmiennej Vstr [nr] -– otwiera okno wprowadzające tekst do zmiennej
- Wprowadź datę do zmiennej VDateTime [nr] – otwiera okno wprowadzające datę do zmiennej
- Wprowadź datę i czas do zmiennej VDateTime [nr] – otwiera okno wprowadzające datę i czas do zmiennej
- Skasuj tablicę tekstową [nr] – kasuje tablicę tekstową o podanym numerze
- Wprowadź nazwę pliku do zmiennej Vstr [nr] – otwiera okno wyboru pliku

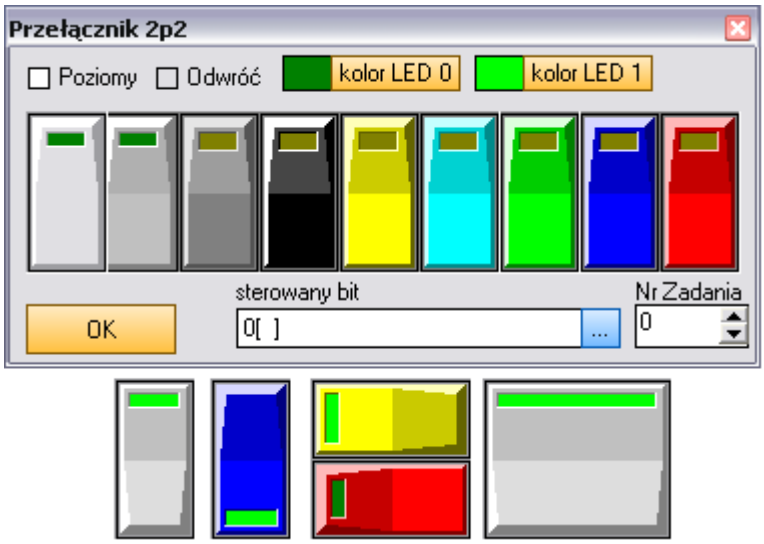
10.16 Komponent Przełącznik dwu pozycyjny 1



Komponent przełącznik dwu pozycyjny steruje stanem wybranego bitu.

Parametry komponentu:
Poziomy – określa czy przełącznik ma być w poziomie czy w pionie
Odwróć – zamienia miejscami strony przełącznika
Tekst 0, Tekst 1 – tekst wyświetlany w odpowiednich połówkach przełącznika oraz jego czcionka.
Kolor 0, kolor 1 – kolor odpowiednich połówek przełącznika
Style przełączników – wybierane przez kliknięcie w wybrany przełącznik – zmieniany jest wtedy styl suwaka.
Płaski – zmienia styl przełącznika na płaski, Płaski p – zmienia styl suwaka przełącznika na płaski
Bit sterowany (zmienna Vbit) wybrany z tablicy zmiennych
Nr Zadania – numer zadania task wywoływanego po zmianie stanu przełącznika.

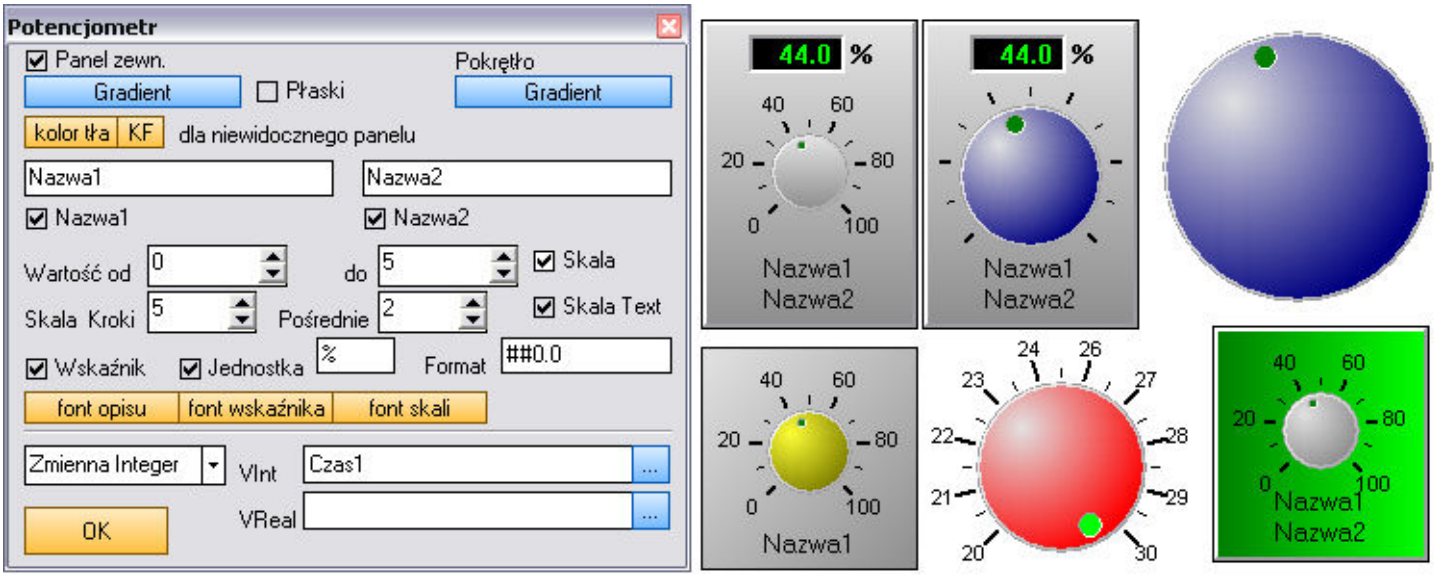
10.17 Komponent Przełącznik dwu pozycyjny 2



Komponent przełącznik dwu pozycyjny steruje stanem wybranego bitu.

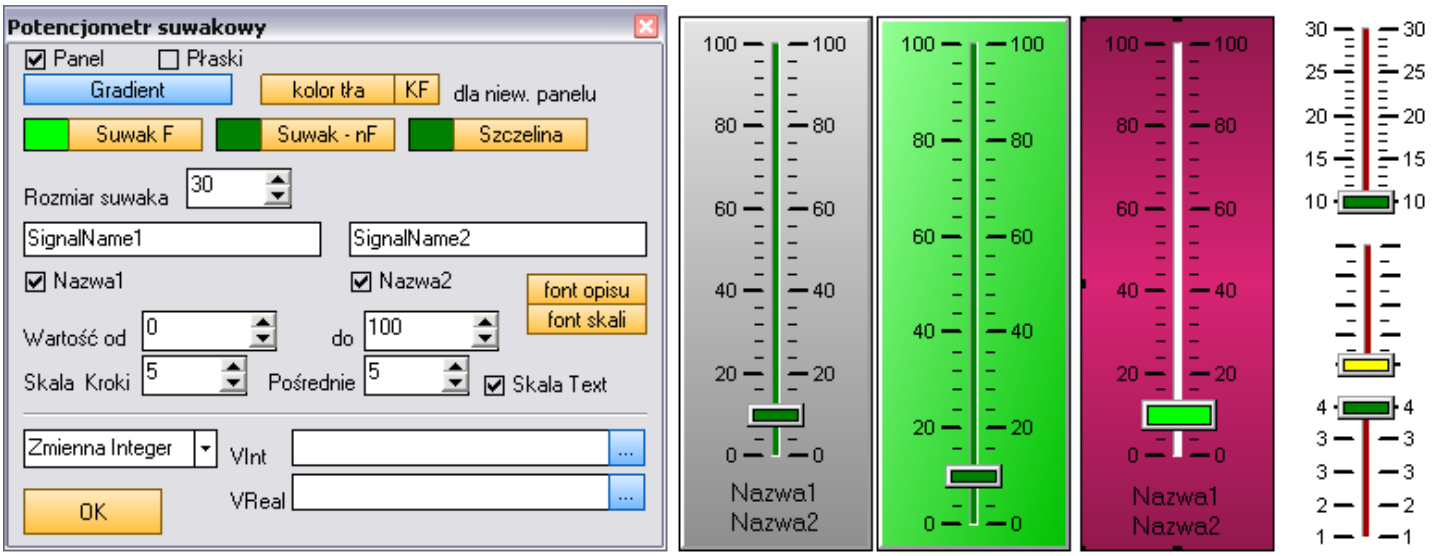
Parametry komponentu:
Poziomy – określa czy przełącznik ma być w poziomie czy w pionie
Odwróć – zamienia miejscami strony przełącznika
Kolor LED 0, kolor LED 1 – kolor kontrolki przełącznika dla odpowiedniego stanu.
Style przełączników – wybierane przez kliknięcie w wybrany przełącznik
Bit sterowany (zmienna Vbit) wybrany z tablicy zmiennych
Nr Zadania – numer zadania task wywoływanego po zmianie stanu przełącznika.

10.18 Komponent Potencjometr



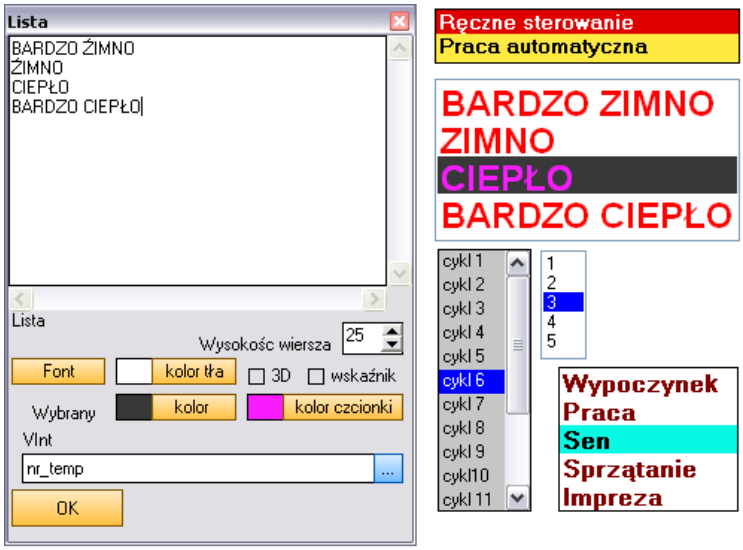
Komponent potencjometr steruje wybraną zmienną typu Vint lub Vreal.
Parametry komponentu :
Kolory i styl gradientu panelu zewnętrznego, kolory i styl gradientu gałki, widoczność panelu zewnętrznego i jego płaski styl.
Kolor tła kiedy panel zewnętrzny jest niewidoczny. Przyciskiem KF można wpisać kolor formularza na którym umieszczono potencjometr.
Dwie nazwy i ich widoczność.
Zakres zmian wartości oraz parametry skali: ilość kroków, ilość kroków pośrednich, widoczność, widoczność tekstu skali i font skali.
Widoczność wskaźnika numerycznego, jego jednostkę i format. Parametr format określa sposób wyświetlania liczby- ilość zer przed i po przecinku, czy mają być wyświetlane nieznaczące zera. Jeżeli ustalimy format na „####0.00" to liczba 345.678 będzie wyświetlona jako 345,67 a dla maski „000000.0" wyświetlone zostanie 000345.8. Uwaga – jeśli liczba może przyjąć wartość ujemną to żeby znak został wyświetlony musi na początku być #
Font opisu i font wskaźnika
Przełącznik rodzaju zmiennej (Integer lub Real) oraz adres zmienna do której zostanie zapisana wartość ustawiona potencjometrem

10.19 Komponent Potencjometr suwakowy



Komponent bardzo podobny do poprzedniego. Różnicą jest to że zamiast gradientu pokrętła ustawiamy rozmiar suwaka, kolor szczeliny oraz kolor suwaka dla komponentu będącego i nie będącego w „fokusie" (czyli będącego aktywnym)

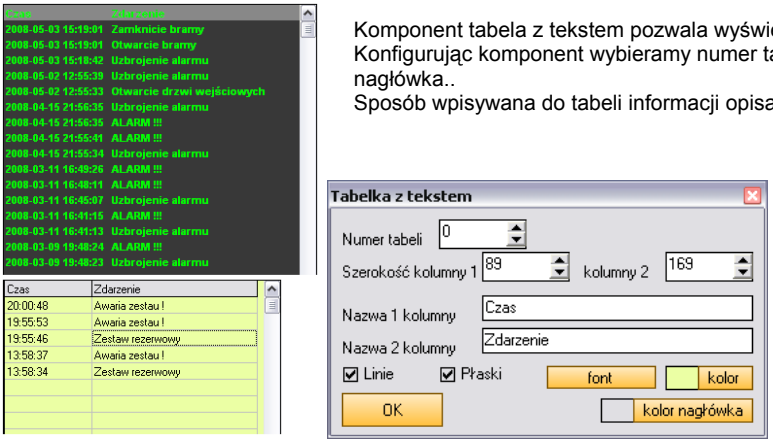
10.20 Komponent Lista



Komponent lista może pracować zarówno jako wskaźnik jak i jako przełącznik. Wybrana pozycja listy powiązana jest z wybraną zmienną typu integer w ten sposób że pierwsza pozycja to zero, druga jeden itd.
Kliknięcie w pozycję zmienia wartość zmiennej. Można zablokować możliwość zmiany pozycji wtedy będzie ona wskaźnikiem.

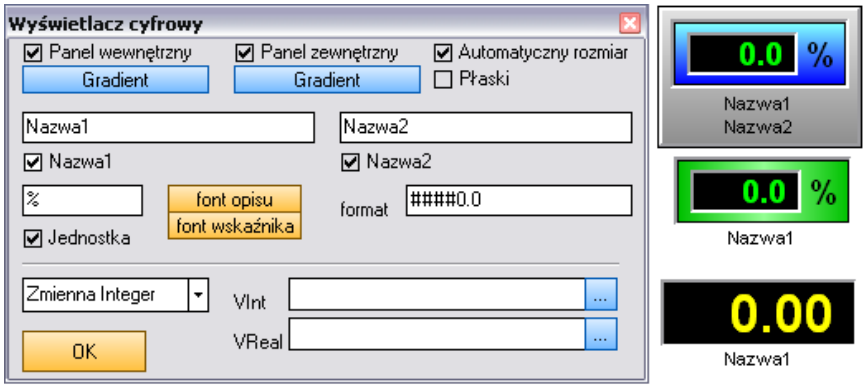
Parametry komponentu:
Lista wierszy
Czcionka listy i wysokość wiersza
Kolor tła
Kolor tła wybranej pozycji i czcionki na wybranej pozycji

10.21 Tabela z tekstem



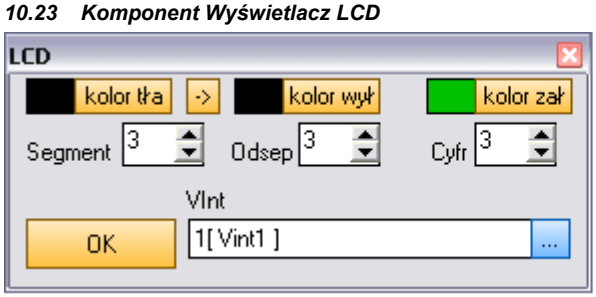
Komponent tabela z tekstem pozwala wyświetlać jedną z ośmiu tablic tekstowych dostępnych w programie. Konfigurując komponent wybieramy numer tabeli, szerokość i nazwę jej kolumn, czcionkę, kolor tła i kolor nagłówka..
Sposób wpisywana do tabeli informacji opisano przy okazji opisu rozkazów do tego służących.

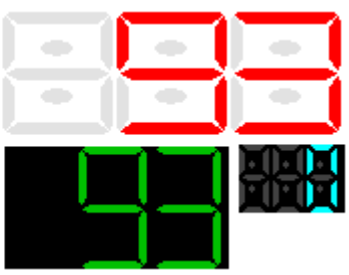
10.22 Komponent Wyświetlacz cyfrowy



Komponent wyświetla wartość wskazanej zmiennej typu integer lub real.
Parametry komponentu:
Styl gradientu i widoczność panelu zewnętrznego
Styl gradientu i widoczność panelu wewnętrznego
Automatyczny rozmiar – załączenie powoduje automatyczne dostosowanie wymiarów do wielkości czcionek.
Teksty wyświetlane pod wskaźnikiem, ich widoczność i czcionka
Czcionka wskaźnika, jednostka i format wyświetlanej liczby.

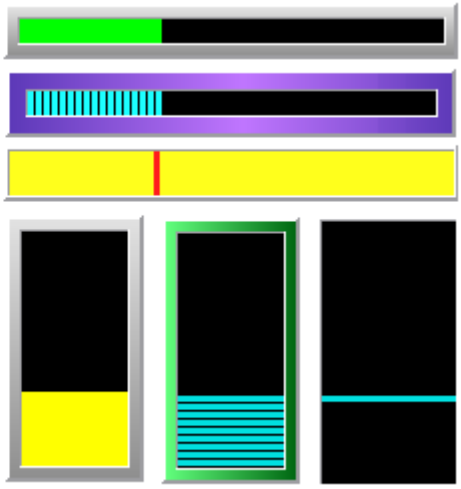
10.23 Komponent Wyświetlacz LCD





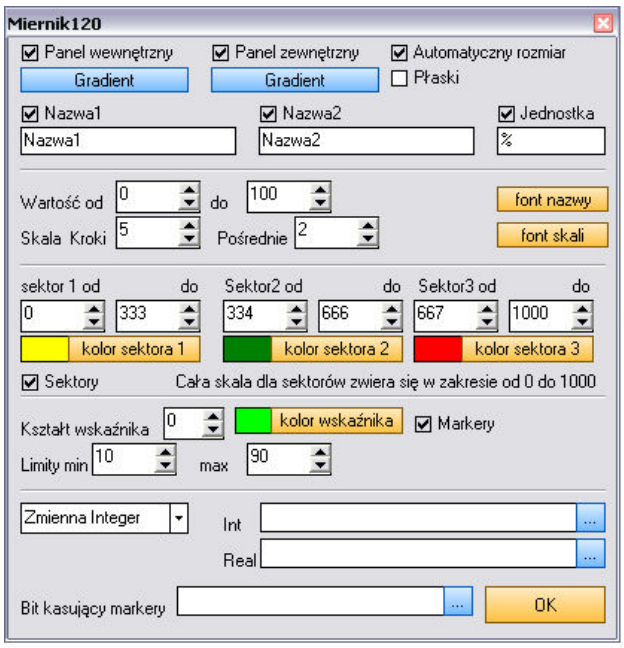
Komponent wyświetla zmienną typu Vint.
Parametry komponentu:
Kolor tła, kolor wyłączonego segmentu kolor załączonego segmentu. Przycisk -> powoduje skopiowanie koloru tła do koloru wyłączonego segmentu.
Szerokość segmentu, odstęp między cyframi i ilość cyfr.

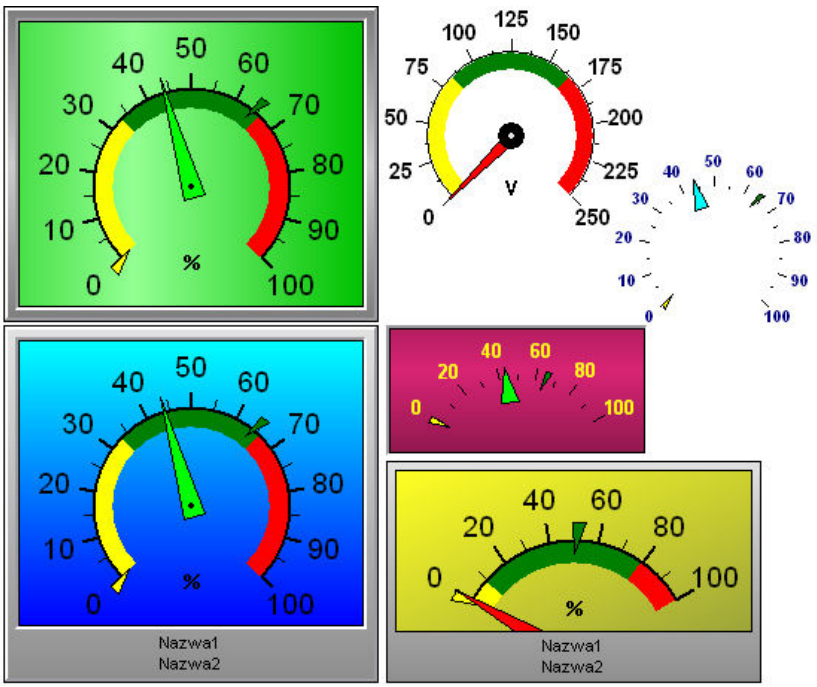
10.24 Komponent BarGraf



Komponent bargraf przedstawia wartość zmiennej integer lub real jako słupek.
O tym czy bargraf jest poziomy czy pionowy decydują proporcje jego wymiarów. Jeżeli wysokość jest większa od szerokości to bargraf działa w pionie, inaczej w poziomie. Bargraf może mieć formę słupka, linijki led lub wskaźnika.
Pozostałe parametry to:
Kolor słupka i tła, gradient panelu i jego szerokość. Ujemna szerokość spowoduje że panel (ramka otaczająca bargraf) zniknie całkowicie.
Wartość Od Do reprezentowana przez bargraf

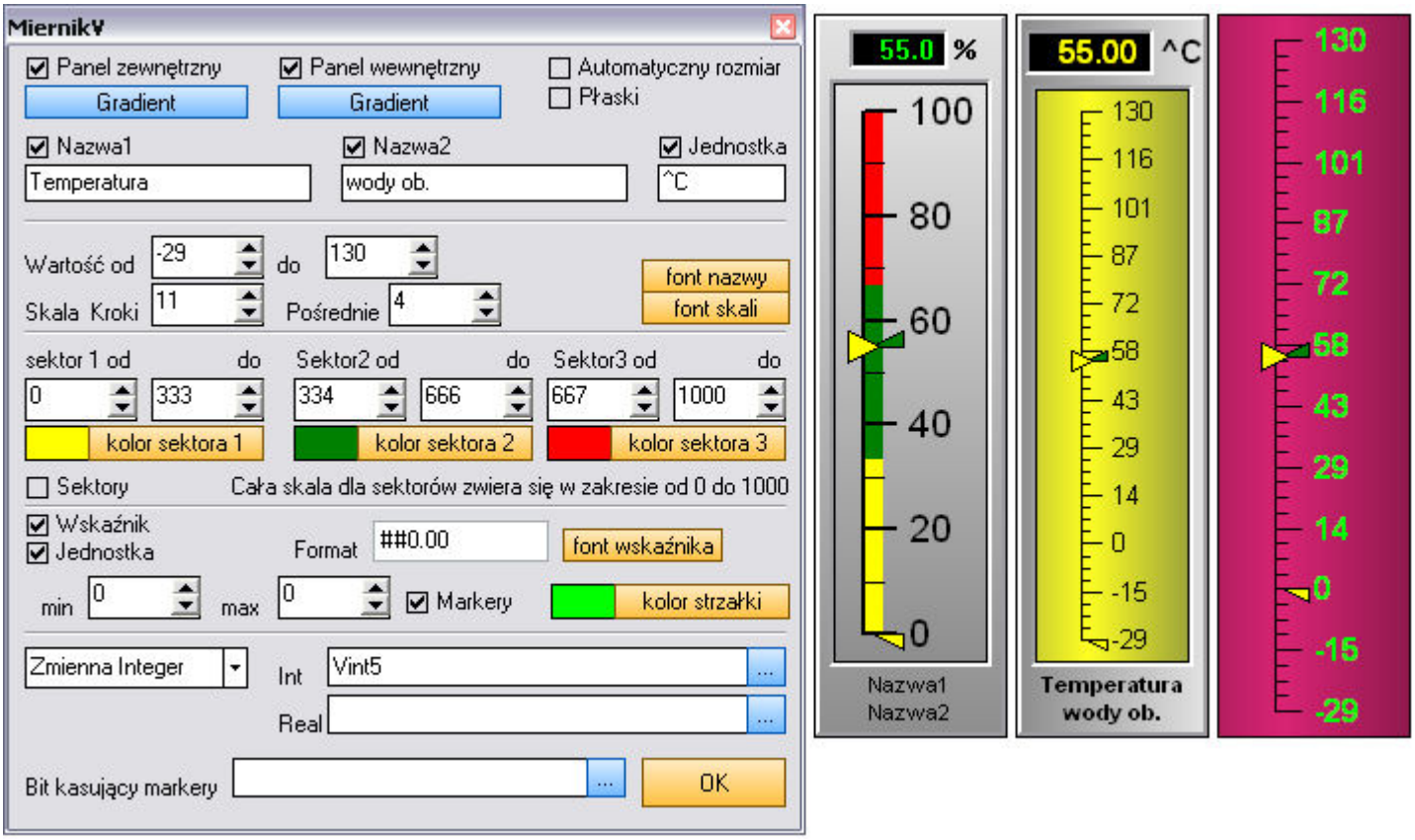
10.25 Komponenty Miernik120 i Miernik270





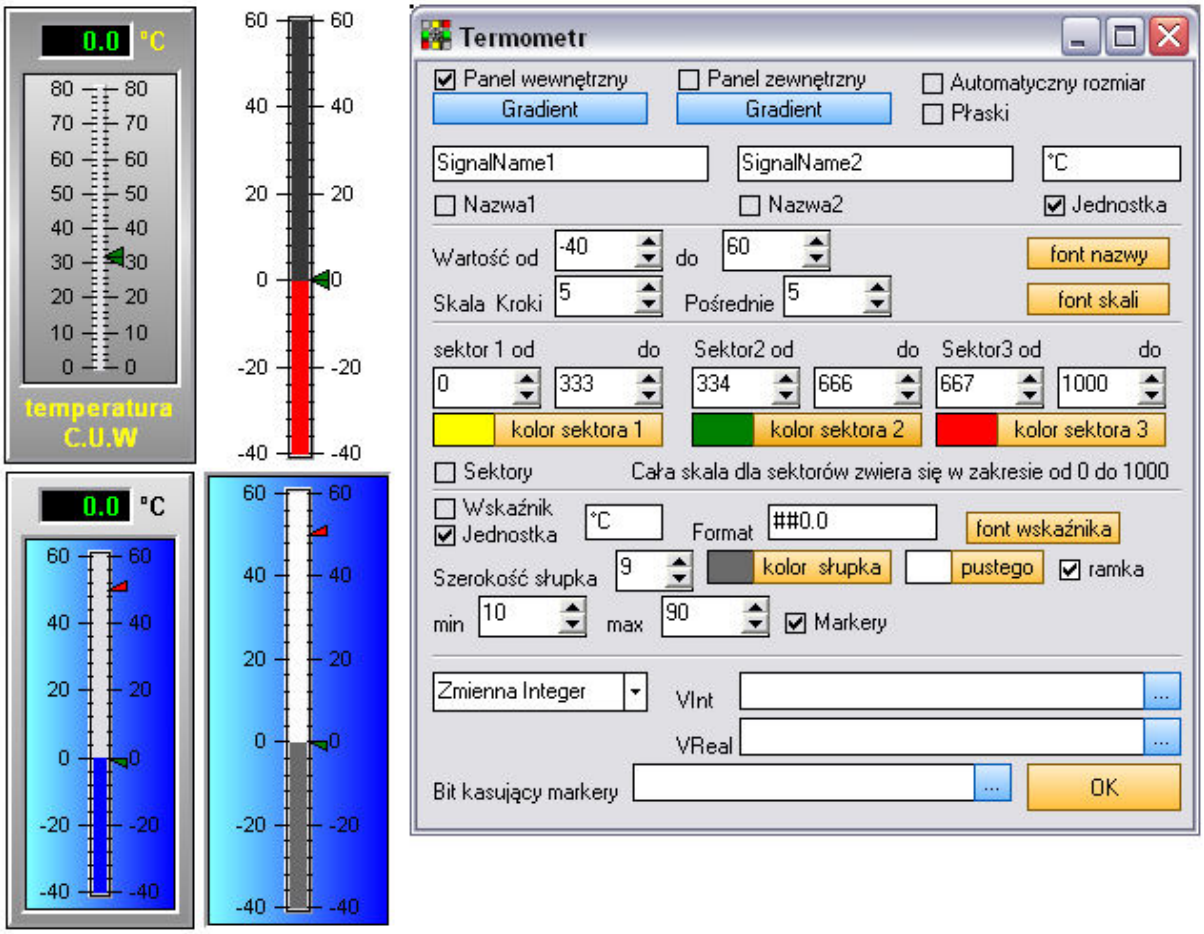
Komponenty miernik120 i miernik270 są identyczne i różnią się tylko kontem skali
Parametry komponentu:
kolory i styl oraz widoczność panelu zewnętrznego, kolory i styl oraz widoczność panelu wewnętrznego, płaski styl paneli
Dwie nazwy, jednostka i ich widoczność.
Wartość zakresu od do, Ilość kroków i kroków pośrednich skali, font nazwy i font skali.
Sektory (zobacz niżej)
Kształt i kolor wskaźnika (strzałki), widoczność markerów limit górny i dolny przekroczenia.
Kiedy wartość limitu przekroczy wartość (górną lub dolną) wyświetlanego parametru to wskaźnik zaczyna pulsować. Jeżeli nie chcemy aby wskaźnik pulsował należy ustawić limit min < od wartości początkowej i limit max > od wartości końcowej zakresu
Markery (małe strzałki zaznaczające osiągniętą wartość minimalną i maksymalną jaka została osiągnięta) skasować można zmienną Vbit. Dzięki temu możemy jednym przyciskiem albo programowo skasować (ustawić na aktualnej pozycji wskazówki) markery wielu wskaźników.

10.26 Komponent MiernikV



Praktycznie wszystkie ustawienia miernika MiernikV pokrywa się z ustawieniami mierników miernik120 i miernik270

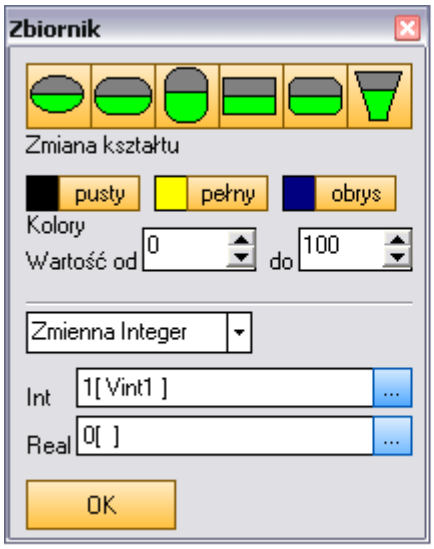
10.27 Komponent Miernik – termometr

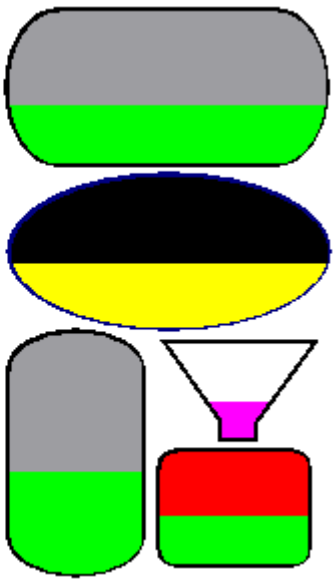


10.28 Mierniki – sektory

Wszystkie mierniki i bargraf maja możliwość podziału skali na trzy sektory o wybranych kolorach. Cała skala to zakres 1000 kroków.
Dla każdej sekcji określamy odpowiedni zakres. Powiedzmy ze chcemy aby pierwsze 20% skali było żółte (1 sektor) ostatnie 30% czerwone (3 sektor) a reszta zielona (2 sektor). Ustawiamy więc zakres od 0 do 200 dla 1 sektora, od 700 do 1000 dla 3 sektora i od 201 do 699 dla sektora drugiego.

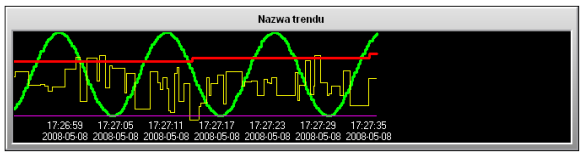
10.29 Komponent Zbiornik

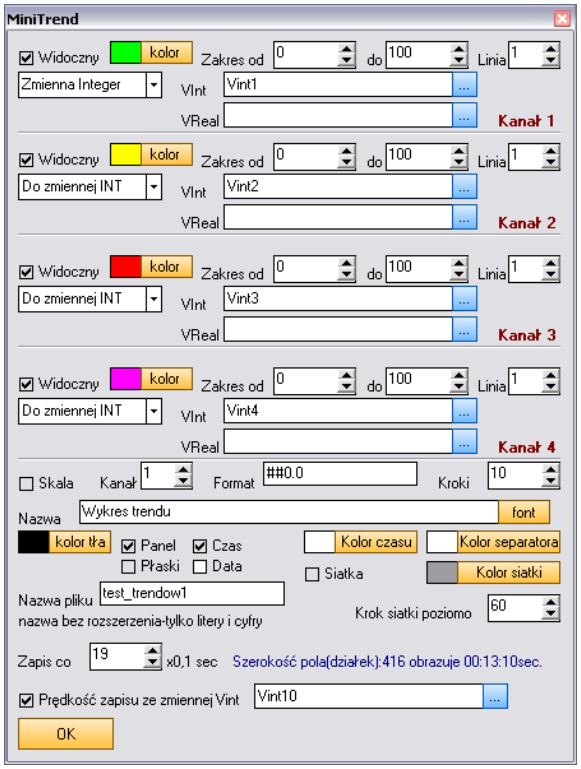




Komponent zbiornik to specjalny bargraf którego kształt może przyjąć kształt charakterystyczny dla zbiornika.
Poza wyborem kształtu dla komponentu ustalamy:
Kolory pustego i pełnego zbiornika oraz kolor jego obrysu
Zakres od wartości do wartości
Zmienną sterującą integer lub real.

10.30 Komponent Mini Trend





Komponent MiniTrend służy do rysowania trendów zmian czterech sygnałów.

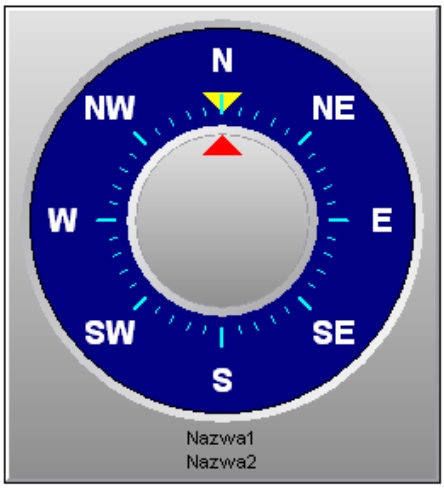
Parametry ustalane dla każdego z czterech kanałów (sygnałów):
Wybór typu zmiennej – integer albo real, zmienna która będzie rysowana, zakres od wartości do wartości, kolor i grubość linii.
Możemy też wyłączyć rysowanie danej linii ograniczając ilość rysowanych zmiennych do 1,2 lub 3 zmiennych

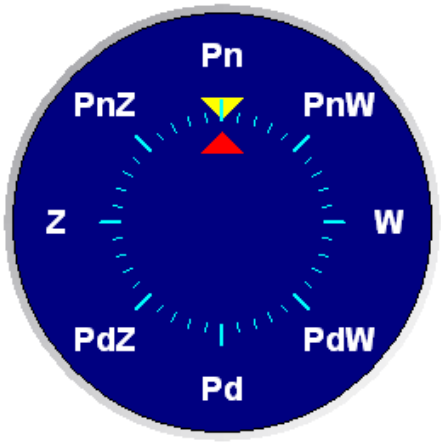
Pozostałe parametry komponentu:
Nazwa i font nazwy, skala – określa czy z prawej strony wykresu ma być widoczna skala, kanał – określa według zakresu którego kanału rozrysować skalę, format wyświetlanej wartości skali.
Kolor tła, kolor czasu wyświetlanego pod wykresem, kolor separatora.
Widoczność daty i/lub czasu pod wykresem, widoczność, ilość kroków i kolor siatki.

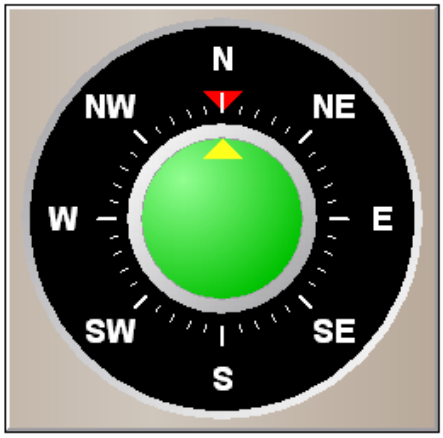
Dane zapisywane są na wykresie co określony czas wyrażony w 10 częściach sekundy. Można powiedzieć że czas zapisu to to czas na jedną działkę (jeden piksel). Czas zapisu może być albo ustawiony na stałe podczas projektowania aplikacji albo może być odczytywany ze wskazanej zmiennej Integer.
O tym jaki okres czasu reprezentowany jest na wykresie decyduje jego długość (długość pola odczytowego) oraz czas zapisu. Podczas ustawiania czasu zapisu obok wyświetlany jest czas jaki pokrywa wykres.

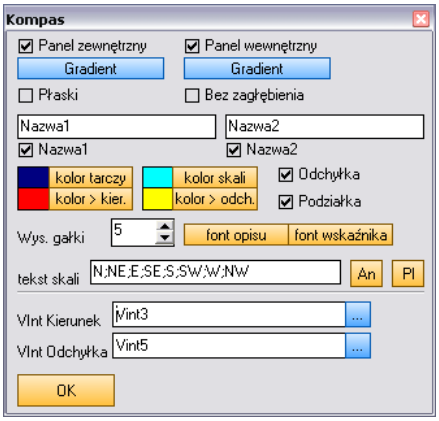
Aby nie utracić zarejestrowanego trendu możemy spowodować aby program zapisywał go sukcesywnie na dysku a po uruchomieniu programu załadował ponownie. W tym celu w polu nazwa pliku wpisujemy nazwę (bez kropki i rozszerzenia) pliku do którego będzie składowany wykres. Należy zadbać aby nazwa pliku była inna dla każdego użytego w aplikacji wykresu.
Plik zapisywany jest do podkatalogu SET. Pionowa biała linia która pojawia się na wykresie po ponownym uruchomieniu programu rozdziela wykres załadowany z pliku od wykresu stworzonego po uruchomieniu aplikacji

10.31 Komponent Kompas









Komponent kompas pokazuje kierunek i odchyłkę od kierunku.
Oby dwie wartości pobierane ze zmiennych integer powinny zawierać się w zakresie od 0 do 359.

Poza parametrami określającymi wygląd komponentu możemy zmienić tekst skali w ten sposób że wpisujemy nowe oznaczenia rozdzielając je średnikami. Przyciski An i PL wpisują opis skali angielski lub polski.

11 Zmienne

11.1 Zmienne globalne i zmienne skryptu

Każda informacja: stan wejścia, stan licznika, tekst, data przechowywana jest w określonej zmiennej.
W systemie Hall zmienne możemy podzielić na dwie grupy – zmienne programu zwane dalej zmiennymi globalnymi oraz zmienne skryptu.
Zmienne programu które będziemy nazywali dalej zmiennymi globalnymi są zdefiniowane na stałe a zmienne skryptu możemy zdefiniować wedle potrzeb wewnątrz skryptu pascalowego.

11.2 Zmienne globalne i ich typy

Każda ze zmiennych użytych w programie ma pewien typ czyli może przechowywać informacje o pewnej strukturze. Stan logiczny (prawda lub fałsz lub jak kto woli zero lub jeden) przechowywane są w zmiennych typu boolean a liczby zależnie od tego czy są całkowite w typach np. integer i real.

Zmienne globalne możemy podzielić na dwa rodzaje: zmienne do dowolnego wykorzystania które nie mają w programie żadnej funkcji i mogą przechowywać dowolne, zgodne ze swoim typem informacje oraz zmienne przypisane do wszelakich urządzeń logicznych i do procesorów

Zmienne uniwersalne :

- **Vbit**1 do **Vbit127** – zmienne bitowe (boolean) mogące przyjmować dwie wartości 1 i 0 W skrypcie operujemy wartościami True i False (prawda i fałsz czyli 1 i 0)
- **Vint** 1 do **Vbit63** – zmienna integer. Liczba całkowita z zakresu od –2147483648 do 2147483647.
- **Vr1** do **Vr63** – zmienna real. Liczba rzeczywista z zakresu od 1.5 x 10^–45 .do 3.4 x 10^38.
- **Vstr1** do **Vstr128** – zmienna tekstowa (string). Zmienna może przechowywać tekst o długości do 255 znaków.
- **Vdt1** do **Vdt16** - zmienna przechowująca datę i czas w specjalnym formacie.

Zmienne urządzeń logicznych:

- zmienne timerów: **T1in** do **T32**, **T1Out** do **T32Out** typu boolean oraz **T1CV** do **T32CV** i **T1PV** do **T32PV** typu integer
- zmienne programatorów czasowych **ProgD1** do **ProgD4** , **ProgW1** do **ProgW4** i **TimerT1** do **TimerT4** typu boolean
- zmienne liczników **C1Out** do **C8Out**, **C1up** do **C8up**, **C1dn** do **C8dn** i **C1res** do **C8res** typu boolean oraz **C1** do **C8** i **C1pv** do **C8PV** typu integer
- zmienne filtrów **Filtr1in** do **Filtr16in** i **Filtr1out** do **Filtr16out** typu integer
- zmienne sekwencera **SQ1** do **SQ8** typu boolean i **SQ_ID**, **SQ1int**, **SQ2int** i **SQ3int** typu integer
- zmienne rejestratora danych analogowych **Logger1_d1** do **Logger1_d4**, **Logger2_d1** do **Logger2_d4**, **Logger3_d1** do **Logger3_d4**, **Logger4_d1** do **Logger4_d4** typu real
- zmienne przelicznika **ADC1in** do **ADC16in** i **ADC1out** do **ADC16out** typu real
- zmienne systemowe typu boolean i integer

Zmienne procesorów (x oznacza numer procesora 0 do 7):

- **Px_in1** do **Px_in24** i **Px_out1** do **Px_out24** typu boolean
- **Px_Ain1** do **Px_Ain5** i **Px_Aout1** do **Px_Aout3** typu integer
- **Px_r1** i **Px_r2** typu real

11.3 Tabela zmiennych, nazwa, nazwa funkcji i opis zmiennej

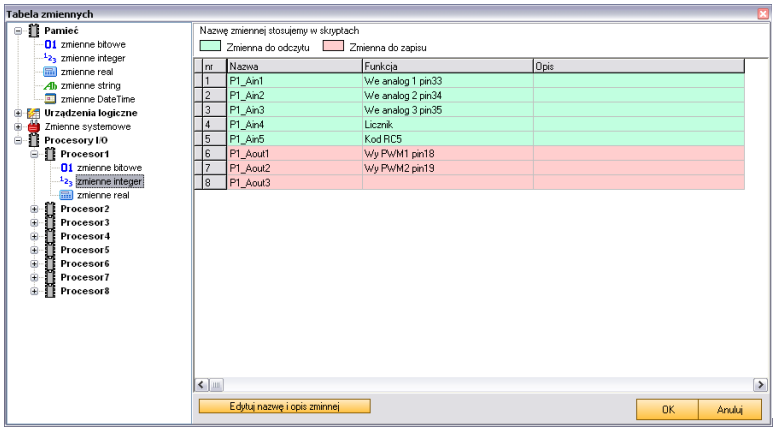


Tabela zmiennych pozwala na wybór zmiennej podczas ustalania parametrów komponentów lub konfigurowania urządzeń logicznych oraz na edycję nazw i opisów zmiennych.

Tabela zmiennych składa się z dwu części – drzewka grup zmiennych i tabeli zmiennych wyświetlającej zmienne z wybranej w drzewku grupy (podgrupy).
Poszczególne kolumny zawierają :

- numer zmiennej
- nazwę zmiennej używaną w skrypcie
- funkcja – opis znaczenia danej zmiennej
- opis – pole pozwalające na dowolne opisanie zmiennej.

Podczas tworzenia programu możemy zmienić nazwę zmiennej na inną, własną . Przykładowo nazwę zmiennej Vbit3 możemy zmienić np. na Drzwi1 aby w programie zamiast pisać

```
if Vbit3 then Vstr10 := 'Drzwi otwarte' else Vstr10 := 'Drzwi zamknięte';
```

zapisać :

```
if Drzwi1 then Vstr10 := 'Drzwi otwarte' else Vstr10 := 'Drzwi zamknięte';
```

możemy też zmienić nazwę zmiennej tekstowej Vstr10 aby zapis programu był jeszcze bardziej czytelny.

Każda zmienna globalna poza nazwą posiada opis funkcji i opis użytkownika . Funkcja opisuje co robi zmienna i dotyczy zmiennych urządzeń i zmiennych procesorów. Funkcje urządzeń zapisane są na stałe a funkcje zmiennych procesorów wpisywane są podczas wyboru typu procesora w konfiguracji sieci procesorów.
Pole opis pozwala na dodanie własnego dowolnego komentarza dla zmiennej.

UWAGA ! Zmiana nazwy zmiennej spowoduje że kompilator skryptu nie rozpozna jej poprzedniej nazwy – jeśli więc zmienimy Vbit3 na Drzwi1 to kompilator nie rozpozna zmiennej vbit3 i zgłosi błąd jeśli ją napotka.

11.4 Typy zmiennych skryptu

W skryptach możemy używać następujące typy zmiennych:

Typ logiczny
Boolean - wartość False (0) lub True (1)

Typy dla liczb całkowitych
Byte zakres od 0 do 255
Word zakres od 0 do 65535
Integer zakres od –2147483648 do 2147483647

Typ dla liczb naturalnych
Real zakres od 5.0 x 10^–324 do 1.7 x 10^308

Typ dla tekstów
String przechowuje tekst do 255 znaków

Typ dla daty i czasu
Tdate przechowuje w specjalnym formacie informację o dacie
Ttime przechowuje w specjalnym formacie informację o czasie
TDateTime przechowuje w specjalnym formacie informację o dacie i czasie

Wyszczególniono tu tylko podstawowe typy zmiennych. Osoby znające język Pascal mogą zastosować w skryptach sterujących wszystkie charakterystyczne dla tego języka typy zmiennych

11.5 Zmienne lokalne procedur i funkcji

Za pomocą dyrektywy Var możemy zdefiniować zmienną lokalną w skrypcie.
Może to być zmienna dla całego skryptu jeśli jej definicję umieścimy na początku skryptu np. :
var licznikx:integer;
Zmienna licznikx będzie widziana we wszystkich zadaniach skryptu.
Można też zadeklarować zmienną lokalną dla danej procedury (zadania) np.

```
Procedure Task1;
var x:integer;
 Begin
   x:= 256 * vint1
   vstr1 := IntToStr(x);
 end;
```

Zmienna x widoczna jest tylko w obrębie procedury i po jej zakończeniu jej zawartość zostanie utracona.

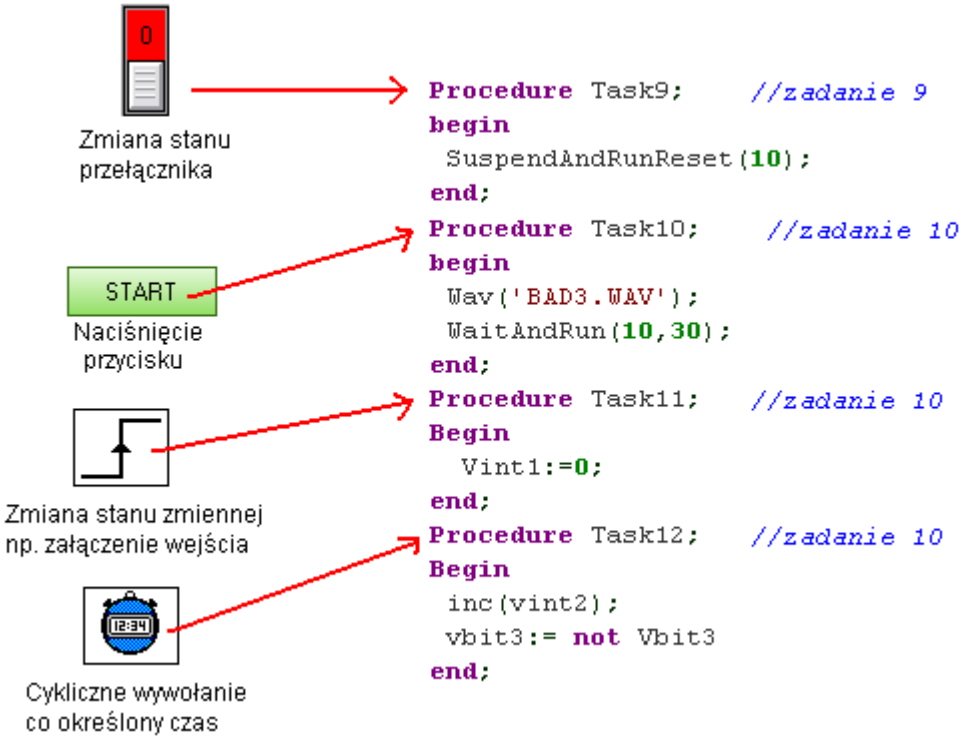
11.6 Zmienne procesorów

Zmienne procesorów odwzorowują w programie aplikacji stan zmiennych w procesorach w ten sposób że poza świadomością programisty stan zmiennych wejść procesorów przepisywany jest do zmiennych wejściowych programie i na odwrót – stan zmiennych wyjściowych przepisywany jest do wyjść procesorów.
Zmienne zostały podzielone na osiem bloków – po jednym bloku danych na procesor.
Ponieważ jednak zmienne procesorów nie mają na sztywno przypisanych funkcji należy je traktować jako rejestry których znaczenie zależne jest od tego jaki procesor został do nich przypisany.
Wynika to z tego że system zaprojektowano w ten sposób aby w przyszłości można było definiować nowe procesory o różnych typach.
Po wyborze procesora w tabeli zmiennych widzimy opis funkcji danej zmiennej dla przypisanego procesora.

Procesory wymieniają dane za pomocą następujących zmiennych (Px oznacza numer procesora) :

- **Px_in1** do **Px_in24** typu bitowego - zmienne do odczytu reprezentujące informację przysyłaną z procesora (wejścia)
- **Px_out1** do **Px_out24** typu boolean – zmienne do zapisu wysyłane do procesora (wyjścia)
- **Px_Ain1** do **Px_Ain5** typu integer - zmienne do odczytu reprezentujące informację przysyłaną z procesora (wejścia analogowe)
- **Px_Aout1** do **Px_Aout3** typu integer - zmienne do zapisu wysyłane do procesora (wyjścia analogowe)
- **Px_r1** i **Px_r2** typu real - zmienne do odczytu (temperatury)

12 Zdarzenie i zadanie



Programowanie systemu oparte jest na obsłudze zdarzeń. Zdarzenie pociąga za sobą wykonanie zadania. Zdarzeniem może być zmiana stanu zmiennej, koniec odmierzania jakiegoś czasu lub naciśnięcie przycisku na ekranie.
Zadanie to procedura w skrypcie sterującym. A więc
wystąpienie zdarzenia pociąga za sobą wykonanie zadania o numerze przypisanym do zdarzenia.
Przykład 1
Na ekranie umieszczamy komponent przycisk a w jego konfiguracji ustawiamy parametr nr. zadania na 10. Każde naciśnięcie przycisku spowoduje uruchomienie procedury Task10 ze skryptu.
Przykład 2
Przypisujemy zmienna do timera w wyzwalaczach. Timer ustawiamy na 5 sekund a zadanie timera na 7. Co 5 sekund wywoływana będzie procedura Task7
Przykład 3
Definiujmy w wyzwalaczu że zmiana stanu zmiennej odpowiadającej wejściu procesora z 0 na 1 ma uruchomić zadnie 11. Kiedy wejście procesora zmieni stan z 0 na 1 (zbocze narastające) to zostanie wykonane zadanie Task11
Przykład 4
W parametrach okna wpisujemy zadanie 1 dla otwarcia ona i zadanie 2 dla zamknięcia okna. Otwarcie okna spowoduje uruchomienie procedury task1 która może np. odczytać ustawienia, na oknie umieszczamy komponenty pozwalające na zmianę ustawień a po zamknięciu okna wywoływana jest procedura Task2 która zapisuje zmiany do pliku.
W systemie jest wiele elementów które inicjują wykonanie zadania.

13 Skrypt sterujący

13.1 Budowa skryptu

Jak powiedziano wcześniej programowanie systemu oparte jest na zdarzeniach i zadaniach. Gdy nastąpi zdarzenie inicjuje ono wykonanie zadania - wywołuje odpowiednią procedurę Task*n* I to właśnie z takich zadań (procedur) składa się program (skrypt).

A wygląda on tak :

```
Procedure Task1;
 Begin
  //rozkazy pierwszego zadania
 end;

Procedure Task2;
 Begin
  //rozkazy drugiego zadania
 end;
```

i tak dalej aż do zdarzenia Task255.

Na końcu skryptu znajduje się sekcja bez nagłówka Procedure Task zaczynająca się słowem kluczowym Begin i kończąca się słowem End z kropką. Instrukcje w tej sekcji wykonywane są po uruchomieniu skryptu:

```
Begin
  //rozkazy wykonywane jednorazowo po uruchomieniu programu
end.
```

Nie musimy w kodzie umieszczać wszystkich 255 zadań ale uwaga :

Jeżeli wywołamy zadanie o numerze dla którego nie będzie w kodzie odpowiedniej procedury o nazwie Task_nr_zadania to nie zostanie zasygnalizowany żaden błąd.

Kolejność zadań w kodzie skryptu nie ma znaczenia.

13.2 Zadania (Task), rozkazy sterujące pracą zadań

Kod zadania opisuje co dane zadanie ma zrobić – coś załączyć, coś wyłączyć, skopiować jakieś dane z jednego urządzenia logicznego do drugiego, wykonać jakieś obliczenia. Powiedzmy że zmienna P1_in1 reprezentuje stan przycisku podłączonego do wejścia modułu a zmienna P1_out2 przepisywana jest do wyjścia do którego podłączyliśmy lampkę. Chcemy za pomocą przycisku zapalać i gasić lampkę – jedno naciśnięcie zapala, następne gasi.
Możemy to zrealizować np. tak: Przypisujemy P1_bit1 do pierwszego wyzwalacza w którym ustawiamy zadanie 3. Teraz każde naciśnięcie przycisku (zmiana stanu zmiennej z 0 na 1) spowoduje zmianę bitu P1_in1, bo każde jego zbocze narastające spowoduje wywołanie procedury Task3.

```
Procedure Task3;
 Begin
  P1_out2 := Not P1_out2;
 end;
```

W procedurze zadania stan bitu P1_out2 zamieniany jest na przeciwny (operator NOT) czyli każde naciśnięcie przycisku (zdarzenie) wywołuje procedurę Task3 (zadanie).

Warto w tym miejscu powiedzieć o bardzo pożytecznej funkcji związanej z obsługą zadań: SUSPEND. Procedura Suspend powoduje zawieszenie możliwości wywołania wybranego zadania na ustalony czas np. suspend(3,5) zawiesi pracę zadania na ok. pół sekundy. (czas wyrażono w dziesiątych częściach sekundy)

Dodajmy suspend do naszego zadania

```
Procedure Task3;
 Begin
  P1_out2 := Not P1_out2;
  suspend(3, 20);
 end;
```

Jaka będzie różnica w działaniu? Otóż po naciśnięciu przycisku zostanie wywołane 3 zadanie i zmieniony stan lampki oraz zawieszone zadanie na dwie sekundy. Czyli przez dwie sekundy możemy sobie naciskać nasz przycisk do woli – nic się nie wydarzy.

Następnym bardzo przydatnym przy tworzeniu zadań poleceniem jest procedura Exit. Exit powoduje pominięcie reszty kodu procedury zadania.
Niech zmienna Vbit2 przechowuje stan wejścia blokady które ma uniemożliwić zmianę stanu naszej lampki. Możemy to zapisać tak:

```
Procedure Task3;
 Begin
  if not Vbit2 then
    Begin
     P1_out2 := Not P1_out2;
     suspend(3, 20);
    end;
 end;
```

lub

```
Procedure Task3;
 Begin
  if Vbit2 then Exit;
  P1_out2 := Not P1_out2;
  suspend(3, 20);
 end;
```

Działanie obu zadań jest identyczne ale w pierwszym przypadku zmiana stanu i zawieszenie zostanie wykonane wtedy gdy Vbit2 ma wartość 0 (false). W drugim przykładzie jeśli Vbit2 ma wartość 1(true) to nastąpi „wyjście” z procedury zadania bez realizacji następnych linii.

Kolejnym rozkazem sterującym pracą programu jest WaitAndRun. Procedura ta zleca wykonanie zadania po upływie podanego czasu.
Powiedzmy że po naciśnięciu przycisku chcemy aby zapaliła się nasza lampka ale żeby po 10 sekundach zgasła. Task3 zapali ją a Task4 zgasi. Task3 wywołamy za pomocą przycisku jak w poprzednim przykładzie a Task4 zostanie wywołany z opóźnieniem przez rozkaz wewnątrz zadania nr 3.

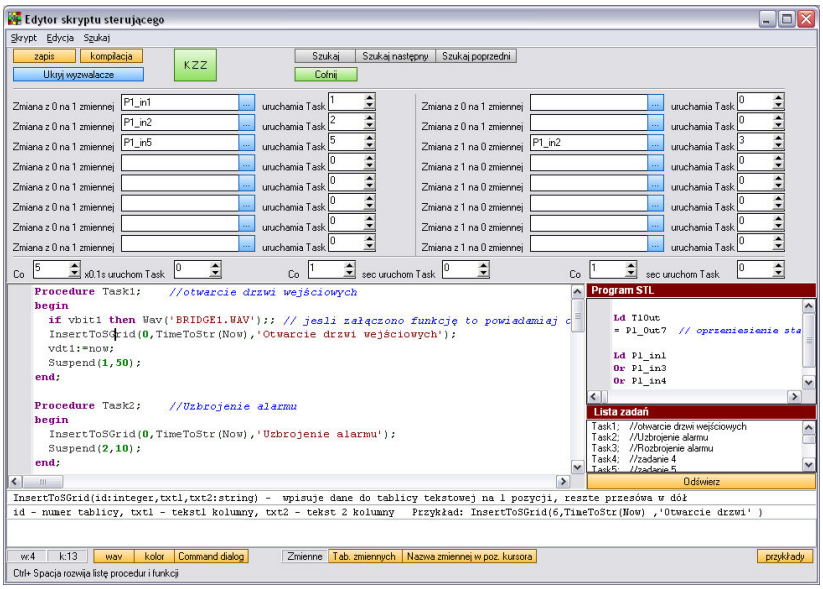
```
Procedure Task3; // zadanie zapalające lampkę
 Begin
  P1_out2 := True;  // zapalamy lampkę
  WaitAndRun(4,100) ; //rozkaz zlecający wywołanie procedury task4 za 10 sekund
 end;
Procedure Task4; // Zadanie gaszące lampkę
 Begin
  P1_out2 := False;
 end;
```

Jeżeli zadanie ma wykonywać się co określony czas możemy je podpiąć do wyzwalacza który uruchamiać będzie je co określony czas.
Możemy też wywoływać je za pomocą procedury WaitAndRun :

```
Procedure Task3;
 Begin
  P1_out2 := not P1_out2;  // zmiana stanu lampki na przeciwny
  WaitAndRun(4,5) ; //rozkaz zlecający wywołanie tej samej procedury za pół sekundy
 end;
```

Raz uruchomione zadanie będzie samo siebie wywoływać co określony czas.
Działanie procedur Suspend i WaitAndRun można przerwać wywołując procedurę SuspendAndRunReset.

14 Edytor skryptu i wyzwalaczy



Okno edytora składa się z następujących elementów:

- edytor wyzwalaczy – można go ukryć przyciskiem zwiń / rozwiń wyzwalacze
- Okno edycji kodu
- Okno podglądu programu STL Okno z listą nagłówków zadań – kliknięcie w nagłówek zadania spowoduje przejście do odpowiedniej linii kodu
- Dwa okienka w których widoczny jest opis rozkazu na którego nazwie stoi aktualnie kursor
- Okno opisu błędów
- Menu górne i dolne

Jednoczesne przyciśnięcie klawiszy Ctrl i Spacja spowoduje otwarcie listy procedur i funkcji.

W górnej części okna dostępne są przyciski :

- kompilacja i zapis kodu
- KZZ – przycisk który kompiluje program a jeśli nie ma błędów to zapisuje go i zamyka okno
- Przyciski wyszukiwania
- Przycisk pozwalający na cofnięcie zmian w kodzie

W dolnej części okna dostępne są przyciski :

- waw – okno dialogowe przeglądarki plików dźwiękowych z możliwością wstawienia rozkazu
- kolor – dialog wyboru koloru – wstawia do kodu zapis szesnastkowy koloru
- Command dialog – okno dialogów generujących komendy Chip_command, TCP_Command i Winamp_command
- Przycisk otwierający okno tablicy zmiennych
- Przycisk otwierający okno tablicy zmiennych i wstawiający do kodu nazwę wybranej zmiennej
- Przycisk otwierający okno podglądu przykładów procedur

14.1 Wyzwalacze

Wyzwalacz uruchamia w pewnych okolicznościach wykonanie określonego zadania. W oknie edytora programu, nad oknem edytorem kodu znajduje się rozwijany formularz z wyzwalaczami. Mamy dwa typy wyzwalaczy: pierwszy reaguje na zmianę stanu z zero na jeden wybranej zmiennej bitowej (np. wejścia), drugi typ wyzwalacza wyzwala zadanie cyklicznie co określony interwał czasu. Mamy 3 wyzwalacze czasowe: pierwszy z podstawą 0,1sekundy i dwa z podstawą sekund.

15 Język STL

Język STL doskonale znany jest programistom sterowników PLC choć w naszym obrazkowym świecie coraz rzadziej jest stosowany.
Język STL zastosowany w Hallu składa się z kilku instrukcji które można podzielić na dwie grupy: jedna grupa rozkazów realizuje operacje na zmiennych bitowych a druga służy kopiowaniu zmiennych integer i real do innych zmiennych integer i real.

Po co STL skoro mamy interpreter Pascala ? Ze względu na szybkość. Kiedy chcemy skopiować stan wejścia analogowego do bloku funkcyjnego służącego do przeliczenia wartości z przetwornika na rzeczywistą wartość pomiaru możemy w skrypcie pascalowym napisać rozkaz ADC1In := P1_Ain1 i umieścić w zadaniu cyklicznie wywoływanym przez wyzwalacz. Ale działanie to będzie realizowane maksymalne 5 razy na sekundę. Możemy też w STLu dodać rozkaz IntToReal P1_Ain1 , ADC1in który wykona to samo kopiowanie. Będzie on wykonywany kilkaset razy na sekundę. Mało tego. W przeciwieństwie do skryptów na których wykonanie program potrzebuje troszkę czasu bo prędkością działania nie oszałamiają program STL realizowany jest bardzo szybko.

W takim samym języku STL choć operującym na innych argumentach i pozbawionym operacji przeniesień zmiennych INT i REAL programuje się mikro sterownik PLC w procesorze HCHPLC

15.1 STL – rozkazy bitowe

Rozkaz ma zawsze jeden parametr będący nazwą zmiennej bitowej, np. Vbit3, T1in etc.
Rozkazy operują na specjalnej zmiennej którą nazwiemy akumulatorem i oznaczymy symbolem ACC. Zmienną dla potrzeb dokumentacji nazywać będziemy Var.

Rozkaz	*Parametr*	*Realizowana funkcja*
LD	var	Kopiuje do akumulatora ACC zmienną Var
LDN	var	Kopiuje do akumulatora ACC zanegowaną zmienną Var
OR	var	Kopiuje do akumulatora ACC wynik operacji ACC OR Var
ORN	var	Kopiuje do akumulatora ACC wynik operacji ACC OR (Not Var)
AND	var	Kopiuje do akumulatora ACC wynik operacji ACC AND Var
ANDN	var	Kopiuje do akumulatora ACC wynik operacji ACC AND (Not Var)
NOT		Neguje stan akumulatora ACC
=	var	Kopiuje stan akumulatora ACC do zmiennej Var

=N	var	Kopiuje zanegowany stan akumulatora ACC do zmiennej Var
SET	var	Jeśli ACC = 1 to ustawia zmienną Var (Var = 1)
RES	var	Jeśli ACC = 1 to kasuje zmienną Var (Var = 0)
BREAK		Jeśli ACC = 1 to pomiń dalsze rozkazy

Jak używać poniższych rozkazów ? Posłużymy się kilkoma przykładami

Przykład 1
Chcemy skopiować stan wejścia 1 procesora do wejścia 1 timera a stan wyjścia 1 timera skopiować do zmiennej Vbit10 i zanegowany do 1 wyjścia

```
LD      P1_in1
=       T1in

LD      T1out
=       Vbit10
=N      P1_Out1
```

Program kolejno:
ładuje do ACC stan we1, zawartość ACC przepisuje do zmiennej T1in
ładuje do ACC stan wyjścia timera, zawartość ACC przepisuje do zmiennej Vbit10 a zanegowaną wartość ACC przepisuje do wyjścia 1

Przykład 2
Chcemy aby zmienna Vbit1 miała stan 1 wtedy gdy dowolne z wejść 1,2 i 3 procesora1 jest ustawiona a zmienna Vbit2 ma być ustawiona tylko wtedy gdy wszystkie wejścia 1,2 i 3 są ustawione. Czyli:

```
Vbit1 :=  P1_in1 OR P1_in2 OR P1_in3
Vbit1 :=  P1_in1 AND P1_in2 AND P1_in3
```

i tak zapisalibyśmy to w skrypcie pascalowym. W STL zapiszemy to w następujący sposób

```
LD      P1_in1
OR      P1_in2
OR      P1_in3
=       Vbit1

LD      P1_in1
AND     P1_in2
AND     P1_in3
=       Vbit2
```

Program kolejno:
ładuje do ACC stan we1, wykonuje sumę logiczną akumulatora i we2, wykonuje sumę logiczną akumulatora i we3, wynik(stan akumulatora) przepisuje do zmiennej Vbit1.
ładuje do ACC stan we1, wykonuje iloczyn logiczny akumulatora i we2, wykonuje iloczyn logiczny akumulatora i we3, wynik(stan akumulatora) przepisuje do zmiennej Vbit2

Przykład 3
Chcemy załączyć wyjście 1 wejściem 1 a wyłączyć wejściem2

```
LD      P1_in1
SET     P1_Out1
LD      P1_in2
RES     P1_Out1
```

Program kolejno:
ładuje do ACC stan we1, sprawdza czy akumulator = 1, jeśli tak to ustawia wy1, ładuje do ACC stan we2, sprawdza czy akumulator = 1, jeśli tak to kasuje wy1

15.2 STL – rozkazy kopiujące zmienne

Poza rozkazami do tworzenia sieci logicznych mamy do dyspozycji cztery rozkazy kopiujące zmienne

rozkaz	zmienna źródłowa	zmienna docelowa	realizowana funkcja
IntToInt	var1 (integer)	var2 (integer)	kopiuje zmienna var1 do zmiennej var2
RealToReal	var1 (real)	var2 (real)	kopiuje zmienna var1 do zmiennej var2
IntToReal	var1 (integer)	var2 (real)	kopiuje zmienna var1 do zmiennej var2
RealToInt	var1 (real)	var2 (integer)	kopiuje zmienna var1 do zmiennej var2 zaokrąglając ja uprzednio.

16 Pascal – język programowania skryptów - podstawy

Jak wspomniano wcześniej skrypty sterujące a w zasadzie zadania zawarte w skrypcie piszemy w języku Pascal.
Osoby znające podstawy tego języka mogą pominąć ten rozdział w którym przedstawimy podstawowe elementy i konstrukcje tego języka.

16.1 Instrukcja przypisania

Przez cały czas pracy programu wszystkie występujące w nim zmienne posiadają określoną wartość. W trakcie wykonywania programu wartość zmiennej można zmieniać wielokrotnie z pomocą instrukcji przypisania:
Zmienna := Wartość

Wyrażenie **:=** nazywamy operatorem przypisania i najczęściej odczytując całą instrukcję określamy je sformułowaniem "Przyjmuje wartość", tzn. instrukcję
Vbit 3 := True
odczytujemy słownie: zmienna vbit3 przyjmuje wartość 1 (wartościom logicznej jedynki i zera odpowiadają słowa True i False czyli prawda i fałsz). Przypisać możemy wartość jednej zmiennej do innej zmiennej np. Vint3 := Vint4, wynik wyrażenia do zmiennej np. Vr1 := Vint3 * 6.84, wynik wyrażenia logicznego: Vbit8 := (Vbit1 and Vbit2) or not Vbit12;

Przypisując wartość jednej zmiennej (wyrażenia) do innej zmiennej musimy pamiętać o zgodności typów – nie możemy przypisać zmiennej całkowitej do zmiennej bitowej – operacja Vbit1 := Vint1 jest nieprawidłowa. Nie możemy też przypisać zmiennej rzeczywistej do zmiennej całkowitej ale możemy przypisać zmienną całkowita do zmiennej rzeczywistej np. Vr3 := Vint12. Znaczenie ma też typ wyniku – jeżeli przypiszemy do zmiennej całkowitej Vint wyrażenie Vint + (-) vint np. Vint2 := Vint4 +23. Ale nie możemy przypisać wyniku dzielenia np. Vint3 := Vint2 / Vint4 bo kompilator nie wie czy wynikiem operacji będzie liczba całkowita czy rzeczywista.

16.2 Operatory i wyrażenia

Operatury arytmetyczne służą do obliczania wartości liczbowych. Wyróżniamy następujące operatory: +, -, *, /, **div** oraz **mod**. Cztery pierwsze z nich chyba nie wymagają komentarza. Operator div to tzw. dzielenie całkowite, czyli dzielenie, a następnie zaokrąglenie wyniku do liczby całkowitej. Operator mod to reszta z dzielenia.

Operatory relacyjne służą do porównywania wartości zmiennych. Wyróżniamy następujące operatory relacyjne:
= równy <> różny < mniejszy > większy <= mniejszy lub równy >= większy lub równy

Operatory logiczne służą do operacji na zmiennych typu logicznego (boolean). Podstawowe operatory to
not - to logiczne "nie" (negacja) - zaprzeczenie twierdzenia
and - logiczne "i" (koniunkcja) - twierdzenie będzie prawdziwe, jeśli wszystkie jego składniki będą równocześnie prawdziwe.
or - logiczne "lub" (alternatywa) - twierdzenie będzie prawdziwe, gdy co najmniej jeden z jego składników będzie prawdziwy.
np. Vbit1:= not Vbit2, Vbit4 := Vbit5 or Vbit6, Vbit10 := Vbit12 and Vbit 14 – możliwe są też wyrażenia złożone: Vbit3 := (vbit5 and Vbit6) or not Vbit20

Operatory logiczne można też stosować dla zmiennych typu integer – wtedy operacje będą przeprowadzane na odpowiadających sobie bitach liczb – np. jeśli chcemy wyzerować bity zmiennej Vint1 tak aby pozostały tylko cztery najmłodsze to możemy wykonać operację:
Vint1 := Vint1 and 15

16.3 Begin, end,i średnik na końcu linii, komentarze

Słowa Begin (początek) i End (koniec) pozwalają na grupowanie instrukcji w jakaś zwartą całość. Wszystkie instrukcje pomiędzy słowem Begin a End traktowane są jako jedna instrukcja. Średnik występujący na końcach niektórych linii, oznacza on zakończenie procedury, funkcji, pętli czy instrukcji warunkowych. Średnika nie stawia się po słowach kluczowych rozpoczynających pewien fragment programu, dopiero stawia się go na końcu tego fragmentu (pomiędzy nimi mogą znajdować się jeszcze inne linie) np. "begin...end;"

Jeżeli chcemy pisać fragment kodu tak aby kompilator pominą ten tekst to oznaczamy go jako komentarz. Jeżeli komentarz ogranicza się do jednego wiersza to przed tekstem wstawiamy dwa ukośniki: // a jeśli chcemy wstawić komentarz zawierający więcej linii to zamykamy go w nawiasy { .. }

16.4 Instrukcje warunkowe IF THEN ELSE

Instrukcja warunkowa pozwala na wykonanie lub zaniechanie wykonania pewnych czynności, w zależności od konkretnego warunku logicznego. Instrukcja ma następującą składnię:

If warunek **Then** instrukcja;

Wykonanie instrukcji polega na sprawdzeniu warunku logicznego i jeżeli jest on prawdziwy, to program wykonuje podaną po słowie Then instrukcję. Jeżeli natomiast warunek logiczny okaże się fałszywy, to instrukcja po słowie Then zostanie pominięta.
Występujący po słowie If warunek może być wyrażeniem relacyjnym: <, >, <=, >=, =, <> lub wywołaniem funkcji o wartościach logicznych, np.

```
if Vbit3 then Vbit3 := false;
Instrukcja if sprawdza czy zmienna Vbit3 ma wartość 1 (true) a jeśli tak to ją kasuje.
```

Warunek logiczny może być również wyrażeniem złożonym utworzonym z prostych wyrażeń logicznych powiązanych operatorami logicznymi języka np.:

```
if Vbit3 and Vbit4 then Vbit5 := false;
if (Vint3 + 100) >= 200 then Vint3 := 0;
```

Po słowie Then może wystąpić tylko jedna instrukcja. Jeżeli chcemy w danym przypadku wykonać więcej poleceń, to należy utworzyć z nich blok - blok traktowany jest bowiem jak jedna instrukcja programu, np.

```
if Vbit3 and Vbit4 then Begin
                         Vbit3 := false;
                         Vstr := 'Jakiś tekst';
                        End;
```

Instrukcja If może mieć postać rozbudowaną:

If warunek **Then** instrukcja1
Else instrukcja2;

Słowo Else oznacza "w przeciwnym wypadku". W tej postaci wykonanie instrukcji polega na wykonaniu instrukcji pierwszej lub drugiej, w zależności od wartości logicznej podanego warunku: jeśli warunek logiczny jest prawdziwy, wykonana zostanie instrukcja1, jeśli fałszywy, program wykona instrukcję2
Wykonanie instrukcji

```
if Vbit3 then Vstr3:='Otwarte' else Vstr3:='Zamknięte';
```

spowoduje przypisanie zmiennej Vstr3 tekstu 'otwarte' gdy vbit3 ma wartość 1 a tekstu zamknięte gdy ma wartość 0. Po słowie Else może wystąpić tylko jedna instrukcja lub blok. W przypadku rozbudowanej postaci ze słowem Else po instrukcji pierwszej nie umieszcza się średnika!

16.5 Instrukcja wyboru CASE

16.6 Pętla For

Są dwie odmiany: pętla idąca w górę i w dół. Składnie odpowiednio wyglądają:

FOR licznik:=wartosc1 TO wartosc2 DO instrukcja;
FOR licznik:=wartosc1 DOWNTO wartosc2 DO instrukcja;

Pętla wykonuje się tyle razy ile wynosi różnica wartości1 i wwartości2 np. For n:= 3 do 15 zostanie wykonana 12 razy. Po każdym obiegu tej pętli zmienna licznik będzie zwiększona (zmniejszona) o jeden.
Po słowie Then może wystąpić tylko jedna instrukcja. Jeżeli chcemy w danym przypadku wykonać więcej poleceń, to należy utworzyć z nich blok - blok traktowany jest bowiem jak jedna instrukcja programu, np.

```
For n:=1 to 10 do
 Begin
  //rozkazy do wykonania
 end;
```

16.7 Procedury i funkcje

Procedury służą do zamknięcia pewnych wielokrotnie wykonywanych ciągów czynności w pewną całość "widzianą" przez program pod konkretną nazwą i operującą w danej chwili na konkretnych argumentach.
Procedurę deklarujemy w następujący sposób :

```
Procedure Nazwa_procedury;
 Begin
  //rozkazy do wykonania
 end;
```

Po zadeklarowaniu procedury możemy wykonać ją w innej części programu, np. wewnątrz innej procedury poprzez podanie jej nazwy.
Zadania które są podstawą programowania systemu Hall to procedury o ściśle określonej nazwie składającej się ze słowa TASK oraz numeru zadania.

Procedura może posiadać parametry z którymi zostanie wywołana i które będą mogły zostać użyte wewnątrz tej procedury.
Powiedzmy że mamy dwa zadania które realizują ten sam ciąg czynności, na przykład po zmianie stanu odpowiedniej zmiennej (wywołują je odpowiednie wyzwalacze) wpisują odpowiedni komunikat do tablicy tekstowej, do dziennika a ponadto otwierają okno z informacją o zdarzeniu.

```
Procedure Task1;
 Begin
  //rozkazy pierwszego zadania
  InsertToSGrid(0,TimeToStr(now),'Otwarcie bramy');
  ToLog(0,1,'Otwarcie bramy');
  AlarmBox('Otwarcie bramy',false,true);
  Suspend(1,50);
 end;

Procedure Task2;
 Begin
  //rozkazy drugiego zadania
  InsertToSGrid(0,TimeToStr(now),'Zamknięcie bramy');
  ToLog(0,1,'Zamknięcie bramy');
  AlarmBox('Zamknięcie ',false,true);
  Suspend(2,50);
 end;
```

Widzimy że oba zadania realizują bardzo podobne czynności. A co jeśli takich zadań będzie osiem? Będziemy osiem razy pisać podobny kod? Możemy więc dla nich zdefiniować procedurę która będzie robić tę część która podobna jest dla wielu zadań. Nazwijmy naszą procedurę EV i przekazujmy do niej parametr o nazwie tekst:

```
Procedure EV(tekst:string);
 Begin
  InsertToSGrid(0,TimeToStr(now),tekst);
  ToLog(0,1,tekst);
  AlarmBox(tekst,false,true);
 end;
```

Teraz zadania będą wyglądały w następujący sposób:

```
Procedure Task1;
 Begin
  //rozkazy pierwszego zadania
  EV('Otwarcie bramy');
  Suspend(1,50);
 end;

Procedure Task2;
 Begin
  //rozkazy drugiego zadania
  EV('Zamknięcie bramy');
  Suspend(2,50);
```

```
end;
```

Pamiętajmy jednak aby procedurę EV zdefiniować (czyli po prostu napisać) nad procedurami zadań.

Funkcja jest podobna do procedury, różni się od niej tym, że pod swoją nazwą zwraca pewną wartość. Funkcja może nie mieć parametrów ale musi zawsze w definicji mieć określony typ zwracanego wyniku.

17 Lista procedur i funkcji stosowanych w skryptach

Poniżej znajduje się lista funkcji i procedur które można używać w skryptach wraz z ich krótkim opisem.

17.1 Funkcje i procedury sterujące programem

Procedury z tego działu zostały dokładnie opisane przy okazji opisu konstrukcji skryptu

```
procedure Suspend(nr_task,delay:integer) ;
```

Procedura blokuje działanie zadania o wskazanym numerze na podany czas w milisekundach
nr_task – numer zadania które ma zostać zablokowane
delaty – czas na który zadanie zostanie zablokowane wyrażony w dziesiątych częściach sekundy.

```
procedure WaitAndRun(nr_task, delay:integer);
```

Procedura powoduje wywołanie wskazanego zadania po upływie wstawionego czasu
nr_task – numer zadania które ma zostać zablokowane
delaty – czas na który zadanie zostanie zablokowane wyrażony w dziesiątych częściach sekundy.

```
procedure SuspendAndRunReset;
```

Procedura anuluje wykonanie procedur Suspend i WaitAndRun.

```
procedure Exit;
```

Wywołanie procedury Exit powoduje opuszczenie aktualnie wykonywanego zadania.

```
procedure SetTaskTimer(timer_nr,time:integer);
```

Podczas tworzenia aplikacji możemy skorzystać z wyzwalaczy do uruchamiania zadań. Poza wyzwalaczami reagującymi na zmianę stanu zmiennej mamy trzy wyzwalacze uruchamiające wybrane zadanie co ustalony czas. Procedura SetTaskTimer pozwala na zmianę ustalonego czasu wyzwalana. Dla pierwszego timera podajemy czas w dziesiątych częściach sekundy a dla drugiego i trzeciego w sekundach.

17.2 Funkcje i procedury matematyczne

17.3 Funkcje i procedury związane z tekstem

```
function IntToStr(i:integer):string
```

Funkcja konwertuje liczbę typu Integer na tekst

```
function UpperCase(s:String):string
```

Wynikiem funkcji jest tekst podstawiony jako zmienna s z zamienionymi wszystkimi literami na duże litery

```
function LowerCase(s:String):string
```

Wynikiem funkcji jest tekst podstawiony jako zmienna s z zamienionymi wszystkimi literami na małe litery

```
function NameCase(s:String):string
```

Wynikiem funkcji jest tekst podstawiony jako zmienna s ze zmienioną pierwszą literą na dużą pozostałe na małe

```
function Length(s:String):integer
```

Wynikiem funkcji jest długość tekstu podstawionego jako zmienna s. Przykładowo wynikiem działanie funkcji

```
function Length('Hall2007')
```

będzie wartość 8

```
function Copy(s:String;start,n:integer):string
```

Wynikiem funkcji jest wycięty fragment tekstu podstawionego jako zmienna s. Wycięcie zaczyna się od znaku o numerze start i ma długość określaną parametrem n. Przykładowo wynikiem funkcji

```
function Copy('Hall2007',3,5)
```

będzie tekst 'll200'

17.4 Funkcje i procedury związane z czasem

Specjalny format zmiennej (typ TdateTime) przechowuje informacje o dacie i czasie.

```
function Now:TdateTime
```

Funkcja zwraca aktualną datę i czas

```
function Date:TdateTime
```

Funkcja zwraca aktualną datę

```
function Time:TdateTime
```

Funkcja zwraca aktualny czas

```
function SecAsHMS(s:integer): String
```

Funkcja zwraca czas w sekundach w postaci tekstu GODZINY:MINUTY:SEKUNDY

17.5 Operacje na bitach

```
function TestBit(n,nrbit:integer):boolean
```
Funkcja zwraca stan wskazanego bitu we wskazanej zmiennej.
przykład:
```
Procedure Task12;
 Begin
  if TestBit(vint,3) then exit;
 end;
```

```
function ChangeBit(n,nrbit:integer;state:integer):integer
```
Funkcja zwraca stan wskazanej zmiennej po zmianie wskazanego bitu na stan określony przez parametr state.
przykład:
```
Procedure Task12;
 Begin
  // zmienna Vint30 przyjmuje stan zakodowany 8 zmiennymi typu bitowego
  Vint30 := ChangeBit(Vint30,0, Vbit10);
  Vint30 := ChangeBit(Vint30,1, Vbit11);
  Vint30 := ChangeBit(Vint30,2, Vbit12);
  Vint30 := ChangeBit(Vint30,3, Vbit13);
  Vint30 := ChangeBit(Vint30,4, Vbit14);
  Vint30 := ChangeBit(Vint30,5, Vbit15);
  Vint30 := ChangeBit(Vint30,6, Vbit16);
  Vint30 := ChangeBit(Vint30,7, Vbit17);
 end;
```

Uwaga. Funkcje testują / zmieniają tylko 8 pierwszych bitów zmiennej (bity 0..7)

17.6 Funkcje i procedury związane z dźwiękiem

```
procedure Beep;
```
Procedura generuje krótki sygnał dźwiękowy.

```
procedure Wav(filename:string);
```
Procedura odtwarza plik dźwiękowy w formacie wav umieszczony w podkatalogu WAV.
Dźwięk z pliku odgrywany jest asynchronicznie – oznacza to że program inicjuje odtwarzanie i nie czeka na jego zakończenie. Jeśli podczas odtwarzania dźwięku z pliku procedura wav zostanie wywołana poraz kolejny to odtwarzanie poprzedniego dźwięku zostanie przerwane

Hall potrafi obsługiwać popularny odtwarzacz Winamp. Przygotowano w tym celu kilka funkcji i procedur:

```
procedure WinAmp_command(command:integer);
```
Procedura wysyła polecenie do odtwarzacza. Polecenia:

1	Start odtwarzania
2	Stop – zatrzymanie odtwarzania
3	Stop z wyciszeniem
4	Następna ścieżka
5	Poprzednia ścieżka
6	Załączenie wizualizacji
7	Wyłączenie wizualizacji

```
function WinAmp_TrackTitle:string;
```
Funkcja zwraca tytuł odtwarzanego utworu

```
function WinAmp_TrackTitle(nr_par:byte):boolean;
```
Funkcja zwraca stan dla parametru nr_par równego 0 – czy odtwarzacz działa (jest uruchomiony), 1 – czy odtwarzacz odtwarza

```
function WinAmp_GelLP(nr_par:byte):integer;
```
Funkcja zwraca dla parametru nr_par równego 0 – pozycję w sekundach a dla parametru równego 1 łączną długość utworu;

```
procedure WinAmp_SetVolume(Volume:integer);
```
Procedura ustawia głośność odtwarzacza. Parametr Volume od 0 do 255 gdzie 255 to maksymalna głośność

Ostatnią z procedur związanych z dźwiękiem jest procedura IVO sterująca syntezatorem mowy Expressivo.
```
procedure Ivo(tekst:string);
```
Wywołanie tej procedury spowoduje wysłanie tekstu do wypowiedzenie do syntezatora mowy Expressivo. Program w żaden sposób nie sprawdza czy syntezator Expressivo jest zainstalowany i aktywny a jeśli nie to nie będzie żadnej reakcji na polecenie.
Uwaga. Pierwszy rozkaz może zostać wykonany z dość dużym opóźnieniem tzn wysłany tekst usłyszeć możemy po kilku sekundach.

17.7 Pamięć podręczna i plik ini

Pewne ustawienia tworzonej aplikacji można zapisać w pliku ini. Może to być np. ścieżka do uruchamianego w aplikacji programu lub pliku – jej umieszczenie w kodzie skryptu mogło by powodować problemy podczas przenoszenia gotowego projektu na inny komputer.
Plik ini umieszczamy w katalogu projekty i nadajemy mu identyczną nazwę jak plik projektu tyle że z rozszerzeniem ini. Plik ini możemy założyć i edytować za pomocą menadżera projektów .

```
function GetIntFromIni(key,parametr:String):integer
```
Wynikiem funkcji jest wartość parametru z klucza key o wartości numerycznej całkowitej. Jeżeli funkcja nie znajdzie parametru, klucza lub wartość będzie nieprawidłowa to wynikiem będzie wartość 0.

```
function GetBoolFromIni(key,parametr:String):boolean
```
Wynikiem funkcji jest wartość parametru z klucza key o wartości logicznej. Jeżeli funkcja nie znajdzie parametru, klucza lub wartość będzie nieprawidłowa to wynikiem będzie wartość 0 (false).

function GetTxtFromIni(key,parametr:String):string
Wynikiem funkcji jest wartość parametru z klucza key o wartości tekstowej. Jeżeli funkcja nie znajdzie parametru, klucza lub wartosc będzie nieprawidłowa to wynikiem będzie pusty tekst (‘’).

przykład:
```
Begin
  Vint30 := GetIntFromIni('Ustawienia','czas1');
  Vbit10 := GetBoolFromIni('Ustawienia','autozapis');
  Vstr11 := GetTxtFromIni('Ustawienia','adreswww');
end.
```

Jeżeli chcemy w trakcie pracy aplikacji zapisać a następnie odczytać jakieś dane to możemy skorzystać z pamięci podręcznej będącej zbiorem tablic o różnych typach zapisywanych do pliku.
Pamięć podręczna to zbiór 1000 elementowych tablic - po jednej dla każdego typu.

procedure SaveMem
Procedura zapisuje pamięć podręczna na dysk
procedure LoadMem
Procedura odczytuje pamięć podręczną z dysku

Zawartość pamięci podręcznej zapisywana jest w pliku o nazwie takiej jak nazwa projektu poprzedzonej zwrotem ‘M_’ i rozszerzeniem .dat w podkatalogu SET

procedure SetIntToMem(id,value :integer)
Procedura zapisuje wartość typu integer do pamięci p0dręcznej.
id – numer w tablicy, value – wartość do zapisu

function GetIntFromMem(id:integer):integer
Wynikiem funkcji jest wartość parametru z pamięci podręcznej. Zwracany jest parametr z elementu o numerze wskazanym przez parametr ID

17.8 Alarmy, tablica tekstowa, dziennik i rejestrator danych

Wszelkie alarmy, ostrzeżenia itp. można wyświetlić za pomocą specjalnego okienka dialogowego o którego treści i momencie pojawienia się decyduje rozkaz:
procedure Alarmbox(txt: String;red,show:boolean)
Procedura dodaje tekst do okna alarmów a następnie powoduje wyświetlenie tego okna.
txt – tekst do wyświetlenia red – jeśli True to tekst będzie wyróżniony
przykład:
```
Procedure Task12;
 Begin
  Alarmbox('Otwarto drzwi',true,true);    // doda i wyświetli wyróżniony tekst  po wykonaniu zadania 12
 end;
Procedure Task13;
 Begin
  Alarmbox('Otwarto okno',false,false);    // doda tekst ale nie wyświetli okna alarmów
 end;
```

Do wyświetlania komunikatów, zestawień etc. służy osiem dwu wymiarowych tablic tekstowych i obsługujące (pokazujące) je komponenty tabela z tekstem opisane w części opisującej budowę ekranów.

procedure ClearSGrid(id:integer);
Procedura kasuje tablice o numerze wskazanym przez parametr id (0 do 7)

procedure ToSGrid(id,row:integer;txt1,txt2:string);
Procedura wpisuje tekst do kolumn we wskazanej tablicy na wskazanej pozycji
id – numer tablicy (0..7)
row – numer wiersza (0..99)
txt1 – tekst wpisywany do pierwszej kolumny
txt2 – tekst wpisywany do drugiej kolumny

procedure InsertToSGrid(id:integer;txt1,txt2:string);
Procedura wpisuje tekst do kolumn na pierwszej pozycji przesuwając jednocześnie pozostałe wiersze w dół tablicy
id – numer tablicy (0..7)
txt1 – tekst wpisywany do pierwszej kolumny
txt2 – tekst wpisywany do drugiej kolumny

procedure SaveSGrids;
Zapisuje zawartość wszystkich tablic na dysk.
procedure LoadSGrids;
Odczytuje zawartość wszystkich tablic z dysku.

Zawartość tablic zapisywana jest w pliku o nazwie takiej jak nazwa projektu poprzedzonej zwrotem ‘SG_’ i rozszerzeniem .dat w podkatalogu SET

Rozwinięciem tablicy tekstowej jest rejestrator zdarzeń – dziennik. Dziennik to tablica znajdująca się w bazie danych SQL

```
xxxxxxxxxxxx  to log !!!!!!!!!!!!!!
```

17.9 Tworzenie dokumentów HTML, E-mail , wbudowana przeglądarka WWW

Jedną z funkcjonalności systemu Hall jest tworzenie dokumentów HTML które potem mogą być udostępniane jako strony www
Działa to w ten sposób że odpowiednio spreparowany plik html umieszcza się w katalogu HTMLSET. W dokumencie tym stosuje się specjalne znaczniki – symbole zmiennych. Wywołanie procedury CreateHTML powoduje odczyt tego pliku, zmianę znaczników na stan zmiennych i zapis we wskazanym katalogu docelowym. Domyślnie jest to katalog HTMLOUT ale w pliku projekt.ini można przypisać inny katalog.
Do katalogu tego generowane są też pliki jpg będące obrazem trendów (zobacz opis komponentu trend) oraz wybrane grafiki z tabeli grafik za pomocą procedury SaveImage. Pozwala to na zastosowanie tych grafik podczas budowy strony.

procedure `CreateHTML(filename:string )`
Procedura kopiuje plik o nazwie filename i kopiuje do katalogu docelowego

procedure `SaveImage(id:integer;state:boolean;filename:string )`
Procedura umieszcza w katalogu docelowym html obrazek z tabeli grafik w formacie jpg
id – numer obrazka w tabeli grafik
state – wybiera czy ma być to obrazek przypisany do stanu 0 czy 1
filename – nazwa pliku obrazka

Można wysyłać za pomocą programu wiadomości e-mail.

procedure `SendMail(account:byte, adress,source:string );`
Procedura inicjuje wysłanie wiadomości e-mail z konta o wybranym numerze na podany adres. Wiadomość sformatowana jest za pomocą podanego pliku matrycy znajdującego się w katalogu SET
account – numer konta
adress – adres odbiorcy
source – nazwa pliku tekstowego z matrycą wiadomości
przykład:

```
Procedure Task12;
 Begin
  SendMail(1,'neuron@neuron.com.pl','mail1.txt');
 end;
```

procedure `SendHeaderMail(account:byte, adress,text:string );`
Procedura inicjuje wysłanie wiadomości e-mail z konta o wybranym numerze na podany adres. Wiadomość posiada tylko nagłówek o podanej treści
account – numer konta
adress – adres odbiorcy
text – treść nagłówka wiadomości
przykład:

```
Procedure Task12;
 Begin
  SendHeaderMail(1,'neuron@neuron.com.pl', Vstr3 );
 end;
```

W systemie dostępna jest też wbudowana przeglądarka www do której można załadować dokument HTML lub stronę WWW za pomocą polecenia:

procedure `WebBrowserLoad(name,link:string;show:boolean;sch:byte)`
Ładuje do wbudowanej przeglądarki dokument html lub stronę www
name – nazwa okna przeglądarki
link – link do pliku lub strony
show – określa czy po wywołaniu procedury uczynić okno widocznym
sch – numer schematu konfiguracji przeglądarki
przykład:

```
Procedure Task12;
 Begin
  WebBrowserLoad('Neuron' ,'http://www.neuron.com.pl',true,2  );
 end;
```

Z poziomu skryptu sterującego dostępna jest bardzo pożyteczna zmienna path_html_out która zawiera ścieżkę do katalogu HTMLOUT co pozwala na ustalenie ścieżki do pliku znajdującego się w tym katalogu np.:

```
WebBrowserLoad('Raport' ,path_html_out+'raport1.html',true,2  );
```

17.10 Rozkazy bezpośredniego sterowania procesorami HallChip

Program wymienia cyklicznie stan zmiennych opisujących stan wejść i wyjść ze zmiennymi w procesorach reprezentujące te wejścia i wyjścia.
Poza tym można sterować procesorami za pomocą poleceń (procedur). Do dyspozycji mamy trzy rozkazy do bezpośredniego sterowania procesorami :

procedure `Chip_Command(nr_procesora,command,parametr:byte)`
nr_procesora - numer (adres) procesora do którego wysyłamy polecenie
command – numer polecenia do wykonania
parametr – parametr dla niektórych poleceń

Działanie polecenia jest zależne od typu procesora i opisane szczegółowo przy okazji opisu procesorów

procedure `Chip_LCD(nr_procesora:byte;text1,text2:string)`
nr_procesora - numer (adres) procesora do którego wysyłamy tekst
text1 – pierwsza linia wyświetlacza
text2 – druga linia wyświetlacza

Polecenie to stworzono specjalnie dla procesora HCHLCCD i służy do wyświetlania tekstów na podłączonym do tego procesora wyświetlacza alfanumerycznego 2*16 znaków

```
procedure Chip_EEPV(nr_EEPV,parametr:byte)
```
nr_procesora - numer (adres) procesora do którego wysyłamy polecenie
nr_EEPV – numer zmiennej EEPV
paametr – wartość która zostanie wysłana do wskazanej zmiennej

Procesor HCHPLC posiada cztery zmienne EEPV których wartość zapisywana jest do wewnętrznej pamięci eeprom za pomocą tego polecenia. Zmienne EEPV opisano przy okazji opisu procesora HCHPLC

17.11 Sterowanie systemem

```
procedure WinExecute(filename:string )
procedure WinExecuteEx(filename,params:string )
```
Procedura uruchamia plik. Jej działanie jest identyczne do kliknięcia myszą w ikonę e eksplorerze windows – jeśli jest to program to zostanie uruchomiony, jeśli jest to inny plik to działanie zależne będzie od tego jaki program skojarzono w systemie z tym plikiem – jeśli będzie to plik avi to najprawdopodobniej zostanie on odtworzony w MediaPlayerze . Procedura WinExecuteEx posiada dodatkowo parametr (listę parametrów) przykład:

```
Begin
  WinExecute(''c:\wideo\film.avi'');
  WinExecuteEx(''notepad.exe'',''c\:doc.txt'')
end;
```

17.12 Funkcje i procedury manipulujące komponentami

Oddziaływać bezpośrednio z programu możemy na komponenty tekst statyczny, figura i grafika.
Wszystkie te komponenty posiadają parametr ID stanowiący numer (adres) komponentu do którego odnosi się procedura. Może być na różnych formularzach kilka komponentów o tym samym ID – wtedy jeden rozkaz zmienia kilka komponentów jednocześnie

```
procedure SetText(id:integer,txt:string );
```
Procedura zmienia tekst komponentu tekst statyczny o podanym numerze ID.

```
procedure SetTextC(id:integer,txt:string;cl:Tcolor );
```
Procedura zmienia tekst i kolor czcionki komponentu tekst statyczny o podanym numerze ID.

18 Timery, programatory czasowe i harmonogram

18.1 Timery

Do dyspozycji mamy 32 timery (przekaźniki czasowe) z podstawą czasu 0,1 sekundy.
Każdy timer ma wejście i wyjście - bity w obszarze zmiennych, Nastawę(PV) i licznik czasu (CV) - zmienne typu integer
Sposób pracy timerów konfigurujemy w konfiguracji urządzeń logicznych. Ustalamy tryb pracy, oraz czy ładować nastawę (PV) po starcie programu.
Każdy timer może pracować w trybie:

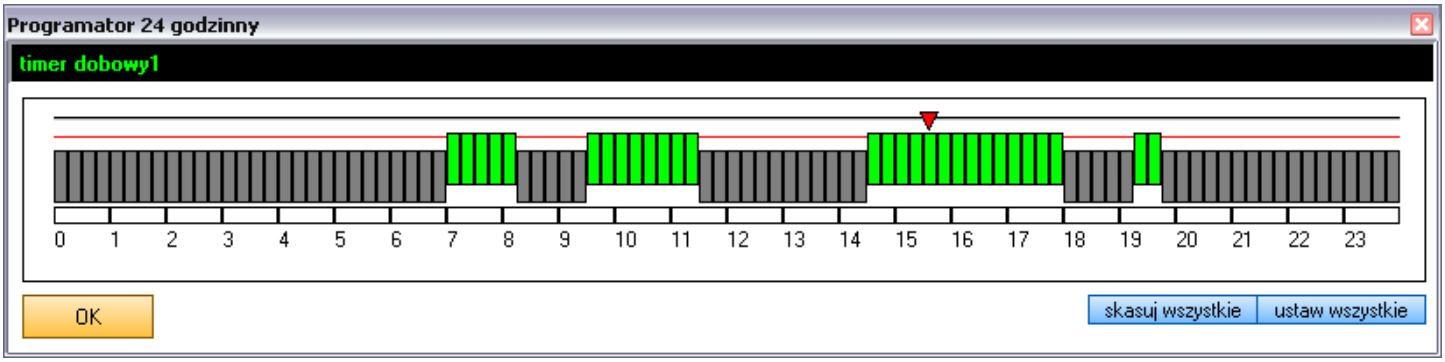
- opóźnienie załączenia – kiedy pojawi się sygnał wejściowy to pojawienie się sygnału wyjściowego opóźnione jest o nastawiony czas. Zanik sygnału wejściowego powoduje zanik sygnału wyjściowego
- opóźnienie wyłączenia – sygnał wyjściowy pojawia się wraz z pojawieniem się sygnału wejściowego a zanika ustawiony czas po zaniku sygnału wejściowego
- impulsu na zboczu – zbocze narastające sygnału wejściowego powoduje wygenerowanie na wyjściu impulsu o ustalonym czasie trwania
- CV > 0 – to specjalny tryb pracy timera. Sygnał wejściowy jest aktywny jeżeli licznik CV jest różny od zera i jednocześnie stan CV jest zmniejszany. W trybie tym inicjujemy timer za pomocą wpisu w skrypcie sterującym np. Ti2CV := 100. Taka instrukcja spowoduje że do licznika timera wpisana zostanie liczba 100, zostanie załączone wyjście timera a po 10 sekundach wyłączone (licznik zostanie zmniejszony do zera)

W programie sterującym do timera możemy odwołać się poprzez zmienne (x oznacza numer timera) :

- TxIn – zmienna typu Vbit - wejście timera
- TxOut – zmienna typpu Vbit – wyjście timera
- TxPV – zmienna typu Vint – nastawa czasu timera
- TxCV – zmienna typu Vint - licznik timera

18.2 Programatory czasowe

Programator dobowy



Cztery programatory dobowe pozwalające na zdefiniowanie sekwencji załączeń / wyłączeń zmiennej obsługiwanej przez dany programator.
Dobę podzielono na 15 minutowe odcinki – klikając myszą w wybraną krzywkę (pomysł programatora wzorowano na popularnych elektromechanicznych programatorach czasowych) podnosimy ją lub opuszczamy. Załączona, zielona krzywka oznacza że zmienna w tym czasie będzie załączona. Można też kliknąć w prostokąt pod krzywkami zmieniając stan wszystkich krzywek danej godziny jednocześnie.
Czerwony wskaźnik nad krzywkami pokazuje aktualną godzinę.
Programatory sterują zmiennymi Progd1, ProgD2, ProgD3 i ProgD4. Okno programatora otworzyć można za pomocą odpowiednio skonfigurowanego przycisku sterującego lub funkcyjnego.
W ustawieniach urządzeń logicznych, na zakładce programatory czasowe definiujemy nazwy wyświetlane w oknach programatorów.

Programator tygodniowy

Programator tygodniowy składa się z 7 programatorów dobowych ustawiających wspólną zmienną zależnie od dnia tygodnia.
Klikając w krzywki odpowiedniego dnia otwiera się programator dobowy pozwalający na zaprogramowanie pożądanej sekwencji.

Czerwony wskaźnik nad krzywkami pokazuje aktualną godzinę a wskaźnik z prawej strony aktualny dzień.
Programatory sterują zmiennymi ProgW1, ProgW2, ProgW3 i ProgW4.
Okno programatora otworzyć można za pomocą odpowiednio skonfigurowanego przycisku sterującego lub funkcyjnego.
W ustawieniach urządzeń logicznych, na zakładce programatory czasowe definiujemy nazwy wyświetlane w oknach programatorów.

Programator Timer

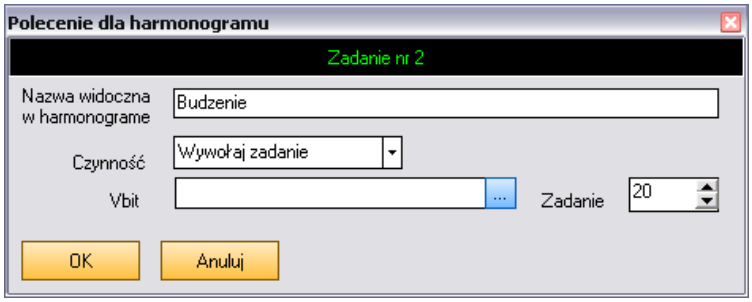
Programator Timer działa na zasadzie popularnego minutnika do jajek. Ustawiamy pokrętło programatora na odpowiedni czas i od tego momentu zaczyna być on odliczany do tyłu.
Podczas odliczania czasu załączona jest zmienna – odpowiednio ProgP1, ProgT2, ProgT3 lub ProgT4. Ponadto po zakończeniu odliczania czasu może zostać wywołane zadanie o zdefiniowanym numerze.

W ustawieniach urządzeń logicznych, na zakładce programatory czasowe definiujemy nazwy wyświetlane w oknach programatorów, zakres czasu – 1 godzina lub jedna doba oraz numer wywoływanego zadania.

18.3 Harmonogram

Programatory czasowe pozwalają na stworzenie sekwencji cyklicznie wykonywanych czynności natomiast harmonogram pozwala na zaplanowanie ich listy – można określić ze dana czynność ma zostać wykonana o takiej to a takiej godzinie takiego to a takiego dnia.

Aby skorzystać z harmonogramu najpierw w oknie konfiguracji urządzeń logicznych wybieramy zakładkę harmonogram i definiujemy polecenie dla harmonogramu. Możemy zdefiniować 30 poleceń które będzie można używać podczas pracy aplikacji

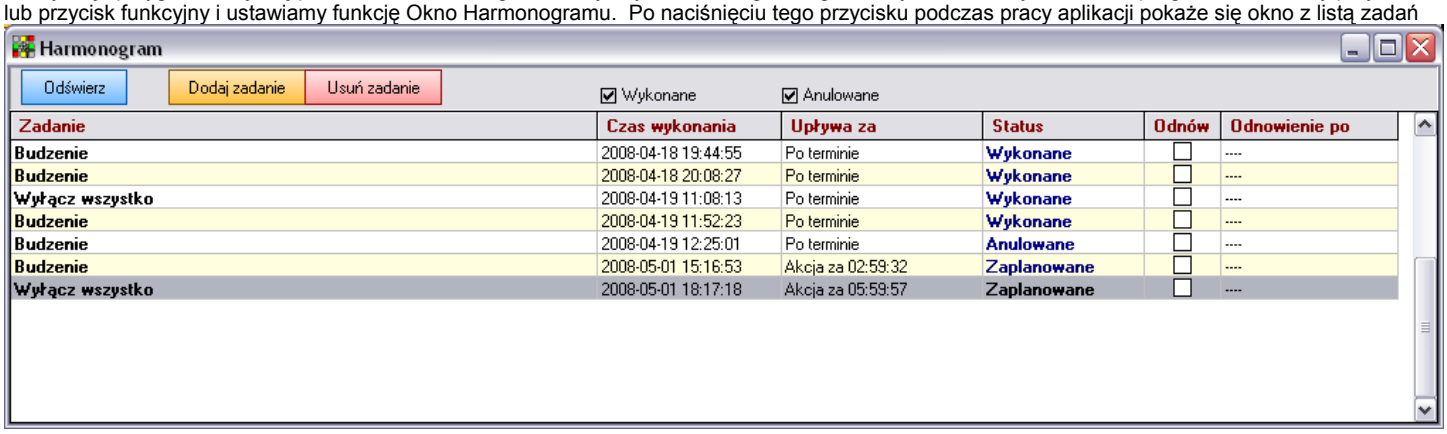


Najpierw definiujemy nazwę która będzie widoczna w harmonogramie gdy użytkownik aplikacji zechce zaplanować jakąś czynność.
Następnie określamy co polecenie ma wykonać. Do wyboru mamy :

- Załączenie wskazanej zmiennej bitowej
- Wyłączenie wskazanej zmiennej bitowej
- wykonanie skryptu o podanym numerze

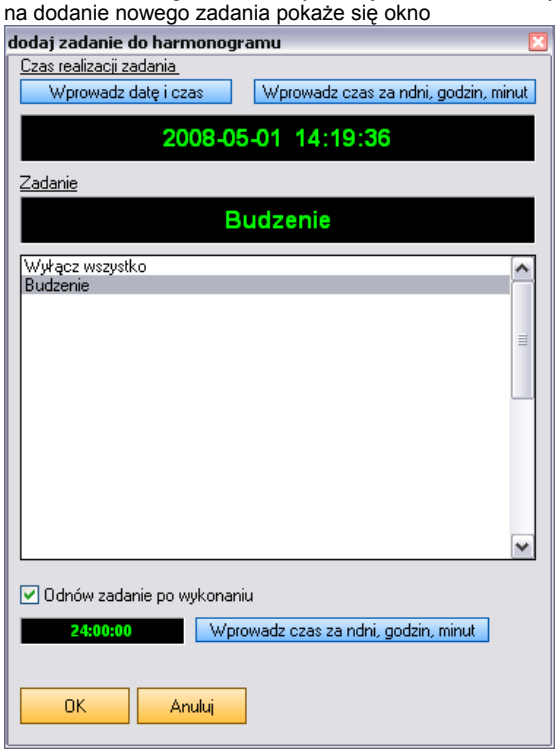
Zależnie od wybranej opcji wybieramy stosowną zmienną albo określamy numer zadania w skrypcie.

Mamy więc przygotowaną listę poleceń dla harmonogramu. Aby użytkownik mógł z niego skorzystać na którymś z okien programu kładziemy przycisk lub przycisk funkcyjny i ustawiamy funkcję Okno Harmonogramu. Po naciśnięciu tego przycisku podczas pracy aplikacji pokaże się okno z listą zadań



Zadanie	Czas wykonania	Upływa za	Status	Odnów	Odnowienie po
Budzenie	2008-04-18 19:44:55	Po terminie	Wykonane	☐	----
Budzenie	2008-04-18 20:08:27	Po terminie	Wykonane	☐	----
Wyłącz wszystko	2008-04-19 11:08:13	Po terminie	Wykonane	☐	----
Budzenie	2008-04-19 11:52:23	Po terminie	Wykonane	☐	----
Budzenie	2008-04-19 12:25:01	Po terminie	Anulowane	☐	----
Budzenie	2008-05-01 15:16:53	Akcja za 02:59:32	Zaplanowane	☐	----
Wyłącz wszystko	2008-05-01 18:17:18	Akcja za 05:59:57	Zaplanowane	☐	----

W oknie harmonogramu widzimy tabelę z zadaniami do wykonania (oraz te zadania które zostaly już wykonane albo anulowane). Jeśli zdecydujemy się na dodanie nowego zadania pokaże się okno



Dodając zadanie określamy kiedy ma być ono wykonane – możemy albo wprowadzić datę i godzinę albo użyć kalkulatorka który nam ją wyliczy po podaniu za ile dni, godzin i minut ma zadanie zostać wykonane.

Następnie wybieramy zadanie z widocznej listy zadań – to właśnie te zadania zdefiniowaliśmy podczas konfiguracji harmonogramu.

Możemy też skorzystać z opcji odów zadanie po wykonaniu. Podajemy wtedy czas przesunięcia dla nowego zadania. Jeżeli załączymy tę opcje to po wykonaniu polecenia automatycznie zostanie dopisane do harmonogramu nowe.

Jeśli więc ustawimy zadanie budzenia na 6:30 następnego dnia a czas wznowienia ustalimy na 24 godziny to po wykonaniu zadania automatycznie wstawi się ono na tę samą godzinę dnia następnego. I tak co dnia aż do czasu usunięcia oczekującego polecenia z opcją wznowienia.

Zadanie może mieć status :

- zaplanowane – zadanie oczekuje na wykonanie bo nie nadszedł jeszcze czas jego realizacji
- wykonane – zadanie zostało zrealizowane zgodnie z planem
- anulowane – program nie działał wtedy gdy zadani powinno zostać zrealizowane – jw. takim przypadku jako przeterminowane zostaje przez program automatycznie ustawione jako anulowane i nie będzie już wykonane

UWAGA!
Harmonogram jest testowany co około dwie minuty.

19 Liczniki i sekwencer

19.1 Liczniki

Do dyspozycji mamy osiem liczników których stan jest dostępny w zmiennych Cx. Stan licznika można zwiększyć lub zmniejszyć zmieniając stan zmiennej CxUp i CxDn – wykrywana jest zmiana z stanu 0 do 1. Licznik można też skasować za pomocą zmiennej CxRes.
Stan wyjścia licznika reprezentowany jest przez zmienną CxOut – jest to zmienna bitowa której stan zależy od trybu w którym ustawiony został dany licznik, stanu licznika Cx oraz stanu nastawy licznika CxPv. Wyjście licznika może pracować w jednym z 4 trybów:

- PV = CV – CxOut = 1 jeśli CxPV = Cx
- PV <> CV – CxOut = 1 jeśli CxPV <> Cx
- PV > CV – CxOut = 1 jeśli CxPV > Cx

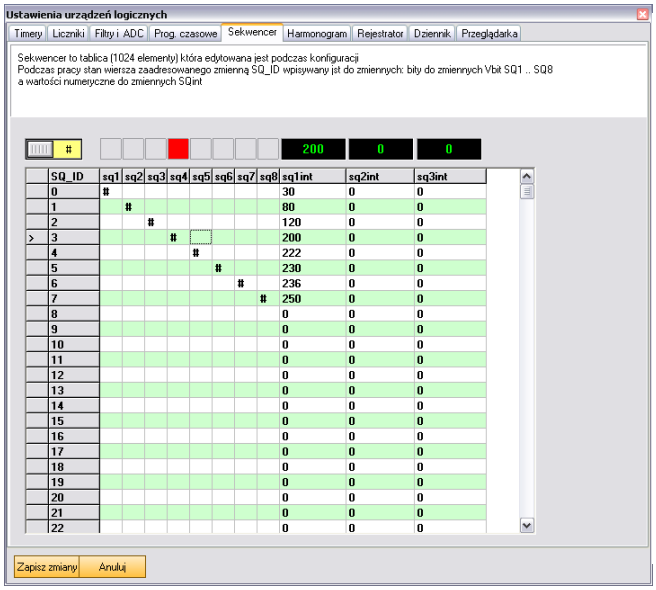
- PV < CV – CxOut = 1 jeśli CxPV < Cx

Nastawa licznika (PV) może zostać automatycznie załadowana ustaloną wartością . Można też konfigurując licznik wymusić kasowanie licznika gdy jego wartość zrówna się z wartością nastawioną PV

W programie sterującym do licznika możemy odwołać się poprzez zmienne (x oznacza numer licznika) :

- CxUp – zmienna typu Vbit - wejście zwiększające licznika
- CxDn – zmienna typu Vbit - wejście zmniejszające licznika
- CxOut – zmienna typu Vbit – wyjście licznika
- CxPV – zmienna typu Vint – nastawa licznika (wartości z którą licznik jest porównywany)
- Cx – zmienna typu Vint - licznik timera

19.2 Sekwencer



Sekwencer to tablica której zawartość edytowana jest w trakcie konfiguracji aplikacji. Każdy wiersz tablicy (tablica ma 1024 wiersze) składa się z ośmiu bitów i trzech wartości numerycznych.

Podczas pracy systemu wartości dane z aktualnego wiersza wpisywane są do zmiennych. Aktualny wiersz to ten którego numer wskazuje zmienna SQ_ID. Stan bitów wpisywany jest do zmiennych SQ1 .. SQ8 a przypisane wartości do zmiennych typu Vint SQ1Int, SQ2Int i SQ3Int.

Za pomocą sekwencera możemy na przykład zrobić sterownie świateł w ten sposób że zadanie wyzwalane co określony czas zwiększy stan zmiennej QS_ID, skasuje ją jeśli przekroczy ilość pożądanych kroków sekwencji a następnie przepisze zmienne bitowe do określonych wyjść.
Każde wywołanie zadania przesunie wskaźnik na następny wiersz.
W bibliotece przykładów szczegółowo opisano sposób sterowania sekwencerem.

20 Obróbka sygnałów analogowych i rejestrator pomiarowy

20.1 Filtry

Często pomierzone wartości analogowe są niestabilne albo zakłócone. Filtry pozwalają na uzyskanie stabilnej wartości i eliminację zakłóceń. Możliwe są dwie metody filtrowania. Albo wynikiem pracy filtru jest średnia z n ostatnich pomiarów albo wynikiem jest wartość n kolejnych odczytów jeżeli są one identyczne. Zmienna wejściowa filtru próbkowana jest co około 200ms.
Dodatkowo możemy ustalić wartość powyżej której pomiar będzie ignorowany (wycinanie szpilek) oraz załączyć maskowanie (zerowanie) najmniej znaczących bitów – można zamaskować jeden lub dwa najmniej znaczące bity.
Do dyspozycji mamy 16 filtrów. Do każdego filtru przypisane są zmienne typu Vint FilterxIn i FilterxOut

20.2 Przeliczniki ADC

Przetwornik analogowo cyfrowy wbudowany w procesor HallChip zwraca, zależnie od napięcia wejściowego i napięcia odniesienia wartość od 0 do 1023 (10 bitów). Powiedzmy że do wejścia przetwornika analogowego podłączyliśmy przetwornik temperatury w taki oto sposób że dla całego zakresu pomiaru temperatury generuje on napięcie pokrywające cały zakres napięcia wejściowego przetwornika - 0-10V dla temperatury 0 - 400 stopni Celsjusza. Powiedzmy że napięcie referencyjne ustawiliśmy na 5V a sygnał wejściowy podzieliliśmy przez 2 dzielnikiem napięcia.
W takim przypadku przy temperaturze 400 stopni uzyskamy z przetwornika ADC wartość 1023. Dla 100 stopni 255, dla 10 stopni 25 itd.
Widać więc że odczyt z przetwornika jest proporcjonalny do temperatury ale tą temperaturą nie jest i wymaga przeliczenia. Do tego właśnie służy blok przelicznika ADC.
W konfiguracji podajemy jakiej wartości analogowej odpowiada wartość zero a jakiej wartości odpowiada wartość 1023 – w naszym przykładzie będą to 0 i 400. Teraz jeśli do zmiennej wejściowej bloku wpiszmy stan przetwornika to w zmiennej wyjściowej trzymamy wartość temperatury.

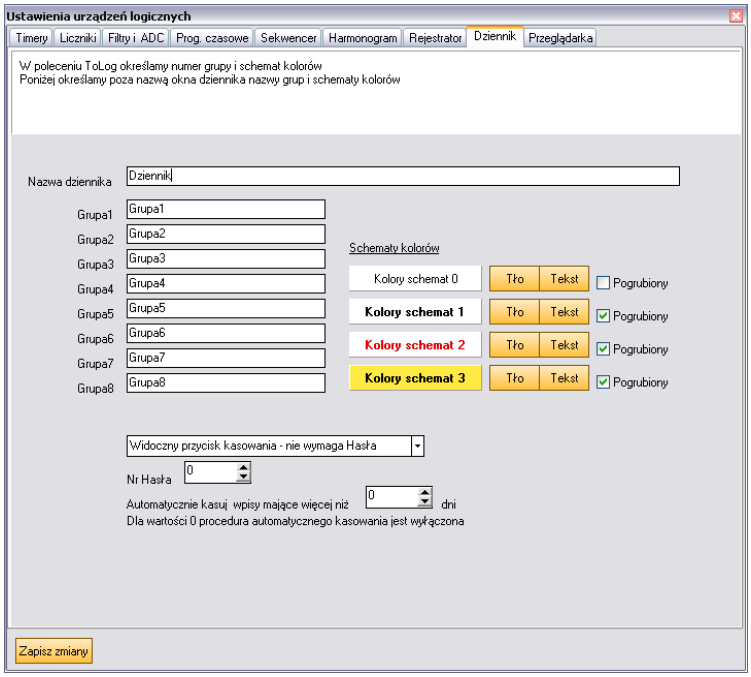
Do dyspozycji mamy szesnaście bloków przeliczających wartości z przetworników pomiarowych. W ustawieniach ustalamy pożądaną wartość wyjściową np. 0 do 100 czyli zakres pomiarowy przetwornika i zakres na który ten zakres zostanie przeliczony. Dodatkowo możemy ustalić jeszcze offset – czyli wartość która zawsze zostanie dodana (odjęta) do wyniku.
Wartości wejściowe i wyjściowe dostępne są w zmiennych typu Vreal: ADCxIn i ADCxOut.

20.3 Rejestrator danych pomiarowych

Poza rejestratorami trendów będącymi komponentami wizualnymi do budowy ekranów opisanymi dokładnie w części opisującej budowę ekranów w systemie dostępny jest rejestrator danych pomiarowych oparty na bazie SQL dzięki czemu możemy gromadzić duże ilości wyników i uzyskać do nich dostęp z wielu programów pracujących w sieci.
Rejestrator gromadzi dane w 4 grupach po 4 kanały. Oznacza to że możemy zapisywać w bazie danych SQL a następnie wyświetlać w formie wykresów i tabel 4 grupy po cztery wartości. Każda grupa i każdy kanał w grupie posiadają swoje nazwy a każdy kanał może zostać indywidualnie wyłaczony.

21 Rejestr zdarzeń

Dziennik to tabelaryczna baza danych przewidziana do przechowywania informacji o różnych zdarzeniach. Wpis do dziennika składa się z dwu części: daty i godziny wpisu oraz treści wpisu.
Wpisu dokonujemy za pomocą instrukcji `ToLog(group, schc:byte,text:string)` np. `ToLog(2,3,'Otwarto drzwi wejściowe');`
gdzie parametr group określa grupę, parametr schc określa schemat kolorów a text to treść komunikatu do wpisania do dziennika
Wpis do dziennika jest oznaczany bieżącym czasem.



Podczas konfiguracji określamy nazwę która będzie wyświetlana w oknie przeglądarki dziennika i nazwy poszczególnych grup

Okno przeglądarki dzienników możemy otworzyć za pomocą odpowiednio skonfigurowanego przycisku określając która grupa ma być widoczna. Jeżeli wybierzemy grupę zero to widoczny w oknie przeglądarki będzie przełącznik pozwalający na wybór grupy lub przeglądanie wpisów wszystkich grup.

Schemat kolorów określa jaką czcionką na jakim tle wpisany zostanie dany komunikat.

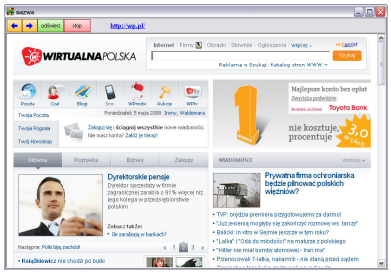
Podczas konfiguracji określamy też sposób kasowania dziennika

- Widoczny przycisk kasowania - nie wymaga Hasła
- Widoczny przycisk kasowania - wymaga Hasła
- Przycisk kasowania niewidoczny

Dla hasła ustalany jest jego numer (priorytet) – zobacz system haseł. Ustalić też możemy automatyczne kasowanie wpisów z dziennika które są starsze niż ustalona ilość dni.

Aby otworzyć w aplikacji okno dziennika należy użyć odpowiednio skonfigurowanego przycisku albo przycisku funkcyjnego. Określamy którą grupe ma wyświetlić

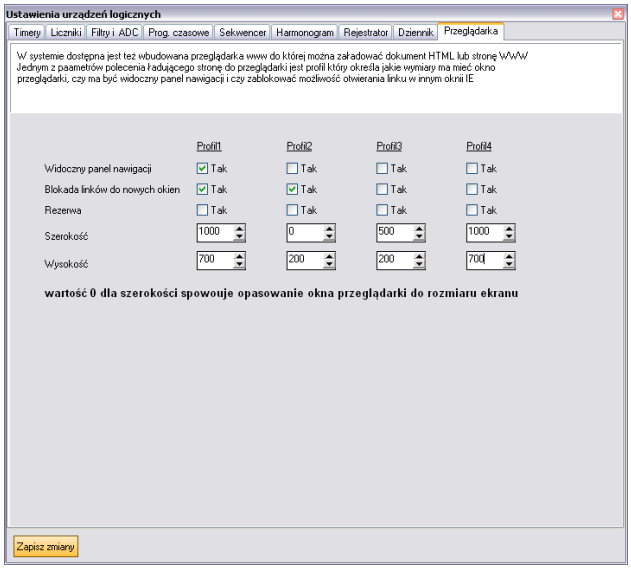
22 Przeglądarka WWW



W systemie dostępna jest też wbudowana przeglądarka www do której można załadować dokument HTML lub stronę WWW za pomocą polecenia:

```
procedure WebBrowserLoad(name,link:string;show:boolean;sch:byte)
```

Ładuje do wbudowanej przeglądarki dokument html lub stronę www
name – nazwa okna przeglądarki
link – link do pliku lub strony
show – określa czy po wywołaniu procedury uczynić okno widocznym
sch – numer schematu (profilu) konfiguracji przeglądarki



przykład:

```
Procedure Task12;
 Begin
  WebBrowserLoad('Neuron' ,'www.neuron.com.pl',true,2  );
 end;
```

Dla każdego profilu możemy ustalić:

- wymiary okna przeglądarki
- czy ma być widoczny panel nawigacyjny (adres, przyciski do przodu, do tyłu, stop i odśwież)
- Czy ma zostać zablokowana możliwość otwierania linków w innym oknie przeglądarki IE (opcja prawego przycisku myszy)

Jeżeli jako szerokość okna zostanie podana wartość 0 to okno zostanie dopasowane do wymiarów ekranu.

23 Hasła dostępu

Przyciski można skonfigurować w ten sposób że otwierają inne okna, zamykają okna, otwierają okna rejestratorów etc.
Ich działanie możemy zabezpieczyć hasłem. Hasła ustalamy za pomocą menadżera haseł wywoływanego przez odpowiednio skonfigurowany przycisk.

Dodając do systemu użytkownika nadajemy mu nazwę, hasło oraz poziom uprawnień będący liczbą od 0 do 5.
Konfigurując przycisk który ma być zabezpieczony hasłem również podajemy poziom dostępu.
Powiedzmy że mamy w systemie 3 użytkowników – jeden ma uprawnienia o poziomie 5, drugi o poziomie trzy a trzeci 1.
Jeżeli przycisk otwierający okno zabezpieczymy hasłem z uprawnieniem na poziomie 3 to otworzy okno tylko użytkownik który ma poziom większy lub równy ustawionemu – czyli pierwszy i drugi użytkownik – trzeci zostanie odrzucony ponieważ jego poziom jest niższy od wymaganego.

24 Wiadomości e-mail

24.1 Konfiguracja kont

Aby wysłać lub odebrać list elektroniczny musimy najpierw zdefiniować parametry konta nadawcy (konta odbiorcy na potrzeby sterowania emailami). Parametry takie jak adres, nazwa konta, hasło i sposób autoryzacji umieszczone są w pliku Mail.ini w podkatalogu Mailer i można je albo edytować ręcznie jak każdy plik ini albo za pomocą przycisku na formularzu ustawień aplikacji wywołać okno konfiguracji kont E-mail.
Ustalamy 3 konta – dwa konta dla procedur wysyłających e-maile i trzecie dla programu sterującego.

24.2 Wysyłanie e-maili

E-maile można wysyłać na dwa sposoby za pomocą instrukcji SendHeaderMail i SendMail. Pierwsza z nich wysyła e-mail bez treści – wysyłamy tylko nazwę (nagłówek) e-maila. Np. SendHeaderEmail(1,'neuron@neuron.com.pl' , 'Testowy list') wyśle na podany adres podany tekst za pomocą pierwszego z zadeklarowanych kont nadawczych. Możemy też użyć zmiennych np.: SendHeaderEmail(vint33, vtxt2 , vtxt3) gdzie w odpowiednich miejscach programu przypisano do zmiennych odpowiednio numer konta, adres odbiorcy i treść nagłówka listu.

Drugie polecenie – SendEmail wysyła na podany adres, za pomocą podanego konta zmodyfikowaną matrycę listu. Jak to działa ?
W podkatalogu SET znajdują się odpowiednie matryce. Plik matrycy to zwykły plik tekstowy. W plikach tych umieszczamy treść listu do wysłania zawierające znaczniki (opis znaczników znajduje się w rozdziale opisującym tworzenie dokumentów html).

Pierwszy wiersz będzie nagłówkiem listu, następne jego treścią.
Powiedzmy że mamy plik raport.txt o następującej zawartości :

```
Raport  temperatur wygenerowany $czas
$Vstr3
$Vstr4
$Vstr6
```

i wykonamy zadanie numer 5 o treści :

```
Procedure TASK5;
 Begin
  Vstr3 := 'Temperatura na zewnątrz  ' + formatfloat('0.00' ,  temp1 )+ '^C';
  Vstr4 := 'Temperatura wewnątrz  ' + formatfloat('0.00' ,  temp1 )+ '^C';
  if  czujnik_drzwi then vstr6 := 'Drzwi otwarte' else vstr6 := 'Drzwi zamknięte';
      SendMail(1 , 'neuron@neuron.com.pl', 'raport.txt') ;
 end;
```

Spowoduje wysłanie na adres neuron@neuron.com.pl listu o nagłówku : Raport temperatur wygenerowany 23-01-2007 14:34:51
o treści :

```
Temperatura na zewnątrz 11.34^C
Temperatura wewnątrz 21.60^C
Drzwi otwarte
```

Czyli program realizując polecenie SendMail pobrał zawartość pliku matrycy 4.txt, zmienił znaczniki na odpowiednie wartości zmiennych i wysłał na wskazany adres za pomocą wskazanego konta.