F&F Filipowski sp.j.
ul. Konstantynowska 79/81
95-200 Pabianice
tel/fax 42-2152383, 2270971
e-mail: fif@fif.com.pl
www.fif.com.pl

Tworzenie aplikacji

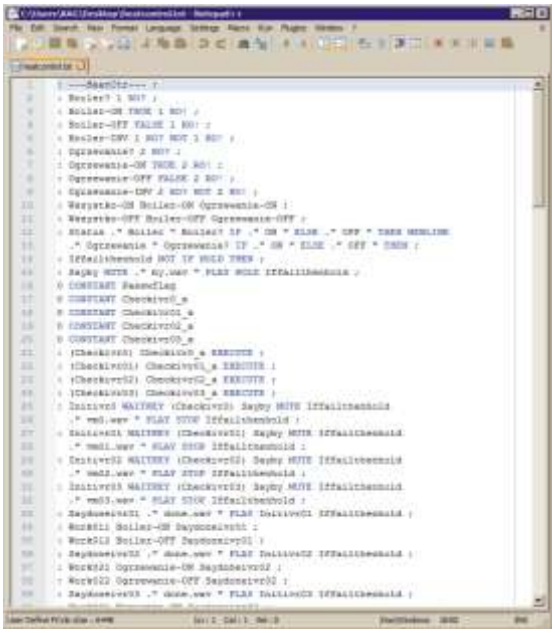
w języku programowania

ForthLogic

Wersja dokumentu A2.0
121128

1 PROGRAMY DO TWORZENIA APLIKACJI

Tworzenie obszernych programów w trybie terminalowym jest trudne i praktycznie się tego nie robi. Ze względu na specyfikę pracy terminalu nie można modyfikować programu. Dlatego tworzenie aplikacji w języku ForthLogic odbywa się w dowolnym edytorze tekstu (np. w Microsoft Notepad, który wchodzi w skład systemu operacyjnego Microsoft Windows XP lub Windows Vista). Należy pamiętać, że długość wiersza nie może przekraczać 77 znaków.
Dla ułatwienia pracy dedykujemy specjalnie do tworzenia aplikacji w języku ForthLogic specjalne środowisko programistyczne Notepad++PuTTY. Program ten jest bezpłatny i jest załączany na płycie CD do każdego sterownika oraz do pobrania na naszej stronie internetowej www.plcmax.pl. Zasady posługiwania się tym programem opisano w osobnej instrukcji.



Notepade++PuTTY

```
( STOPALL FORGET ---HeatCtr---
  ( Zrealizowano proste dwapoziomowe menu glosowe kierowania dwoma
  (wyjsciami przekaznikowymi.
  ( Tymi samymi wyjsciami mozna kierowac przez SMS, za pomoca
  ( przyciskow F1/F2 i za pomoco przyciskow dzwonkowych. Stan wyjsc
  ( mozna obserwowac na wyswietlaczu i na cyfrowych wyjsciach. Przez
  ( menu uzytkownika jest mozliwosc zal/wyl hasla.
: ---HeatCtr--- ;
  ( ================================================================
  ( Definicja Slow (Rowniez dla SMS)
  ( ================================================================
: Boiler? 1 RO? ;
: Boiler-ON TRUE 1 RO! ;
: Boiler-OFF FALSE 1 RO! ;
: Boiler-INV 1 RO? NOT 1 RO! ;
: Ogrzewanie? 2 RO? ;
: Ogrzewanie-ON TRUE 2 RO! ;
: Ogrzewanie-OFF FALSE 2 RO! ;
: Ogrzewanie-INV 2 RO? NOT 2 RO! ;
: Wszystko-ON Boiler-ON Ogrzewanie-ON ;
: Wszystko-OFF Boiler-OFF Ogrzewanie-OFF ;
: Status ." Boiler " Boiler? IF ." ON " ELSE ." OFF " THEN NEWLINE
  ." Ogrzewanie " Ogrzewanie? IF ." ON " ELSE ." OFF " THEN ;
  ( ================================================================
  ( MENU GLOSOWE
  ( ================================================================
  ( sprawdzenie bledow - obecnosci karty pamieci,
  ( podejmowanie wezwania glosowego, i tym podobnie
: Iffailthenhold NOT IF HOLD THEN ;
  ( zakonczenie jadlospisu
: Sayby MUTE ." by.wav " PLAY HOLD Iffailthenhold ;
  (flaga zapytania o haslo w menu glosowym
0 CONSTANT Passwflag
```

Microsoft Notepad

2 ZASADY TWORZENIA APLIKACJI

Poniższy przykład jest praktycznym przedstawieniem budowy aplikacji dla sterownika H04 w oparciu o poznany już zasób słów języka ForthLogic. Pokazuje jak tworzyć obszerny program budując go z poszczególnych mniejszych bloków programowych będących rozwiązaniem części zadania.
W opisach budowanej aplikacji dla wskazania pojedynczych słów lub całych zagadnień, o których mowa będą podawane w nawiasach prostokątnych [ ] numery wierszy, w których się znajdują.

2.1 Początek aplikacji

Dla lepszej przejrzystości programu można dodawać komentarze i opisy poszczególnych fragmentów programu. Służą do tego znaki „(” i „)”. Tekst pomiędzy nawiasami będzie interpretowany przez forth-system jako komentarz i nie zostanie zwrócony jako nieznane słowo. Wszystkie słowa standardowe są pisane dużymi literami, dlatego dla odróżnienia warto definiowane słowa pisać małymi literami .

```
1 ( MÓJ PIERWSZY PROGRAM
2 ( Nazwa projektu: MAX
3 ( data otwarcia projektu: 2010.09.17
4 ( Wersja: v1.0
5
6  STOPALL CLEARSYS
7  2 ERRORLEVEL
8  FORGET program
```

```
9   0 ERRORLEVEL
10 : program ." MAX v1.0 2010.09.17 " ;
```

Tworzymy nagłówek programu, w którym możemy zawrzeć wszelkie informacje na temat budowanej aplikacji, np. nazwa aplikacji, kolejny numer wersji i data, kto jest twórcą, dla kogo jest tworzona i itp. [1-4]. Dla przejrzystości programu zostawiamy jedną linię wolną [5].
Słowa w liniach 6-9 stanowią początek każdej aplikacji. Przy wgrywaniu naszej nowej aplikacji stopują wykonywanie aplikacji będącej w pamięci sterownika (STOPALL), "zerujemy" parametry zasobów systemowych, takich jak zmienne, stałe i stany wyjść (CLEARSYS). Wartość 2 dla słowa ERRORLEVEL pozwala na przyjęcie przez system komendy dla wcześniej nie zdefiniowanego słowa, tutaj "program". Wartość 0 dla słowa ERRORLEVEL przywraca standardowy poziom pracy systemu. Wykasowanie słowa "program" (FORGET) jest wykasowaniem wcześniejszej aplikacji, która zaczynała sie słowem "program".
Następnie definiujemy słowo *program* [11], które otwiera nam aplikację i praktycznie wyznacza początek definiowania nowych słów. Dodatkowo, przy wykonaniu go słowo to wyprowadza do bufora wejściowego tekst, który może kryć nazwę i wersję danej aplikacji. W przypadku, kiedy nie wiemy, jaka aplikacja jest w sterowniku, w łatwy sposób można to sprawdzić wykonując to słowo bezpośrednio w oknie terminalu lub nawet przez telefon komórkowy.

2.2 Blok programowy sterowania wyjściem

Tworzymy funkcję cyklicznego załączania wyjścia DO1.

```
11 ( sterowanie wyjściem DO1
12 : on  1 1 DO! 0.5 3000 BEEP ;
13 : off 0 1 DO! 0.5 1000 BEEP ;
14 : alarm on 1.0 1 TIMER! off 3.0 2 TIMER! alarm ;
15 : stop 0.0 1 TIMER! STOP 0.0 2 TIMER! STOP 0 1 DO! ;
```

Piszemy komentarz opisujący nasz blok programowy [11]. Definiujemy słowa *on* i *off*, które odpowiednio załączają i wyłączają wyjście DO1 z sygnalizacja dźwiękowa wykonania tych słów [12-13]. Słowo *alarm* [14] definiuje na funkcje naprzemiennego załączania i wyłączania wyjścia DO1 w cyklu 1sek/2sek. Słowo *stop* [15] zatrzymuje nam rekursywne wykonanie słowa *alarm* i zeruje wyjście DO1.
Po zdefiniowaniu pierwszych słów możemy już wgrać naszą aplikację. Podając i wykonując słowo *alarm* w oknie terminalu lub przez telefon komórkowy uruchomimy nieskończony cykl załączeń i wyłączeń wyjścia DO1. Cykl ten zatrzymujemy wykonując słowo *stop*.

2.3 Menu funkcji dodatkowych

W poprzednim punkcie nasze zdefiniowane słowa mogliśmy wywołać tylko ręcznie w trybie dialogowym. Dobudujemy teraz blok programowy pozwalający na uruchomienie alarmu wykonując jeden z wierszy funkcji dodatkowych w menu konfiguracyjnym sterownika.

```
16 ( definicja funkcji w menu Funkcji dodatkowych
17 : 1menu ." ALARM " 1 MENU alarm ;
18 : 2menu ." STOP " 2 MENU stop ;
19
20 1menu 2menu
21
```

Jak poprzednio, komentarzem wyróżniamy nowy blok programowy [16]. Definiujemy słowo *1menu* [17], które tworzy wiersz pierwszy z napisem ALARM. Po wybraniu wiersza i naciśnięciu OK uruchomimy słowo *alarm*. Analogicznie słowo *2menu* [18]. Zdefiniowane słowa podajemy w nowej linii [20]. Całość koniecznie zamykamy ostatnią pustą linią [20]. Musimy podać znak ENTER po słowach, aby je wykonać. Analogicznie jak w oknie terminalu. W przeciwnym razie przy wgrywaniu programu do sterownika linia 20 nie zostałaby wprowadzona do forth-sysyemu.

Po wgraniu do sterownika zmodyfikowanej aplikacji słowa *1menu* i *2menu* zostaną automatycznie wykonane i funkcje alarm i stop będą natychmiast dostępne w menu funkcji dodatkowych.

2.4 Automatyczne uruchamianie aplikacji

W poprzednim punkcie stworzyliśmy aplikację, w której zdefiniowane słowa były automatycznie wykonane przy wgrywaniu. Niestety, po wyłączeniu zasilania sterownika nasza aplikacja nie startuje samoczynnie. Słowa zdefiniowane zachowują się w pamięci nieulotnej, ale nie są automatycznie wykonywane. Trzeba zbudować mechanizm, który pozwoli samoczynnie uruchamiać wgraną aplikację.

```
19 ( słowo uruchamiające aplikację: run
20 : run 1menu 2menu ;
21 ." run " BOOT
22 run
23
```

Zamiast bezpośredniego podania słów *1menu* i *2menu* definiujemy nowe słowo *run*. [20]. Słowo to wyznaczamy jako automatycznie „butowane" (ładowane) przy uruchomieniu sterownika [21]. Po restarcie sterownika zostanie wykonane słowo *run*, które wykona słowa *1menu* i *2menu*. Podanie słowa *run* na końcu aplikacji [22] spowoduje wykonanie go natychmiast po wgraniu aplikacji.
Tak zdefiniowany blok programowy praktycznie wyznacza koniec każdej aplikacji.

2.5 Definiowanie funkcji przycisków sterownika

Tworzymy blok programowy, który oprócz menu funkcji dodatkowych pozwoli nasze funkcje *alarm* i *stop* wywoływać za pomocą zdefiniowanych przycisków na czole sterownika.

```
( definicja funkcji przycisków F1 i F2
 : f1 F1 BUTTON alarm ;
 : f2 F2 BUTTON stop ;
```

Definiujemy słowa *f1* i *f2* [20-21]. Musimy je zdefiniować po definicjach słów *alarm* i *stop*, a przed słowem *run*.
Aby były automatycznie uruchamiane przy załączeniu sterownika musimy je dołożyć do definicji słowa *run* [25].

```
 : run 1menu 2menu f1 f2 ;
```

Nasza zmodyfikowana aplikacja wygląda teraz tak:

```
1  ( MÓJ PIERWSZY PROGRAM
2  ( Nazwa projektu: MAX
3  ( data otwarcia projektu: 2010.09.17
4  ( Wersja: v1.0
5
6  STOPALL CLEARSYS
7  2 ERRORLEVEL
8  FORGET program
9  0 ERRORLEVEL
10 : program ." MAX v1.0 2010.09.17 " ;
11
12 ( sterowanie wyjściem DO1
13 : on  1 1 DO! 0.5 3000 BEEP ;
14 : off 0 1 DO! 0.5 1000 BEEP ;
15 : alarm on 1.0 1 TIMER! off 3.0 2 TIMER! alarm ;
16 : stop 0.0 1 TIMER! STOP 0.0 2 TIMER! STOP 0 1 DO! ;
17
18 ( definicja funkcji w menu Funkcji dodatkowych
19 : 1menu ." ALARM " 1 MENU alarm ;
20 : 2menu ." STOP " 2 MENU stop ;
21
22 ( definicja funkcji przycisków F1 i F2
23 : f1 F1 BUTTON alarm ;
24 : f2 F2 BUTTON stop ;
25
26 ( słowo uruchamiające aplikację: run
27 : run 1menu 2menu f1 f2 ;
```

```
28 ." run " BOOT
29 run
30
```

2.6 Sygnały wejściowe - zadziałanie wejść cyfrowych

W większości przypadków potrzeba uruchomić daną funkcję programową w chwili pojawienia się zewnętrznego sygnału sterującego na wejściu cyfrowym. Zdefiniujemy blok programowy, który ustali odpowiednią reakcję sterownika na sygnały wejściowe.

```
( Wej. DI1 i DI2
0 5 FLAG!
0 6 FLAG!
: di5 5 DI? 5 FLAG? NOT AND IF alarm THEN 5 DI? 5 FLAG! 0.1 5 TIMER! di5 ;
: di6 6 DI? NOT 6 FLAG? AND IF stop  THEN 6 DI? 6 FLAG! 0.1 6 TIMER! di6 ;
```

Słowo di5 definiuje nam reakcję na sygnał na wejściu cyfrowym DI5. Słowo *alarm* zostanie wykonane tylko wtedy, gdy pojawi się sygnał na tym wejściu (zbocze narastające). Zanik sygnału na wejściu DI5 nie powoduje niczego. Słowo di6 definiuje nam reakcję na sygnał na wejściu cyfrowym DI6. Ale dla odróżnienia słowo *stop* zostanie wykonane dopiero po zaniku sygnału wejściowego (zbocze opadające). Pojawienie się sygnału wejściowego nie spowoduje wykonania słowa *stop*. Dla rozpatrywania obydwu warunków zastosowaliśmy operator warunkowy IF-THEN. Wykonywanie każdego ze słów zamknięte jest w cyklu 0.1sek, co oznacza, że stan wejścia jest sprawdzany właśnie z taką częstotliwością.
Słowa te należy startować wraz z aplikacją, więc trzeba dołożyć je do definicji słowa *run*.

```
: run 1menu 2menu f1 f2 di5 di6 ;
```

2.7 Zmiana parametrów programu – okno parametryzacji.

Dotychczas nasza funkcja alarm działa zawsze w stałym cyklu 1/2sek. Chcąc dowolnie zmieniać te czasy trzeba zmodyfikować słowo *alarm* oraz dopisać blok programowy wykorzystujący okno parametryzacji do zadawania wartości czasów oraz kolejne punkty menu funkcji dodatkowych do ich wywoływania. Nasza nowa aplikacja wygląda tak:

```
1   ( MÓJ PIERWSZY PROGRAM
2   ( Nazwa projektu: MAX
3   ( data otwarcia projektu: 2010.09.17
4   ( Wersja: v1.0
5
6   STOPALL CLEARSYS
7   2 ERRORLEVEL
8   FORGET program
9   0 ERRORLEVEL
10   : program ." MAX v1.0 2010.09.17 " ;
11   ( definicja stałych t1 i t2
12 1.0 FCONSTANT t1
13 3.0 FCONSTANT t2
14 : def_t1 TOF t1 ;
15 : def_t2 TOF t2 ;
16 : czas_t1 ." t1 [sek] " t1 GET def_t1 STOP ;
17 : czas_t2 ." t2 [sek] " t2 GET def_t2 STOP ;
18 ( sterowanie wyjściem DO1
19 : on  1 1 DO! 0.5 3000 BEEP ;
20 : off 0 1 DO! 0.5 1000 BEEP ;
21 : alarm on t1 1 TIMER! off t2 2 TIMER! alarm ;
22 : stop 0.0 1 TIMER! STOP 0.0 2 TIMER! STOP 0 1 DO! ;
23 ( definicja funkcji w menu Funkcji dodatkowych
24 : 1menu ." ALARM " 1 MENU alarm ;
25 : 2menu ." STOP " 2 MENU stop ;
26 : 3menu ." Czas t1> " 3 MENU czas_t1 ;
27 : 4menu ." Czas t2> " 4 MENU czas_t2 ;
28
29 ( definicja funkcji przycisków F1 i F2
30 : f1 F1 BUTTON alarm ;
```

```
31 : f2 F2 BUTTON stop ;
32
33 ( Wej. DI1 i DI2
34 0 5 FLAG!
35 0 6 FLAG!
36 : di5 5 DI? 5 FLAG? NOT AND IF alarm THEN 5 DI? 5 FLAG! 0.1 5 TIMER! di5 ;
37 : di6 6 DI? NOT 6 FLAG? AND IF stop  THEN 6 DI? 6 FLAG! 0.1 6 TIMER! di6 ;
38
39 ( słowo uruchamiające aplikację: run
40 : run 1menu 2menu 3menu 4menu f1 f2 di5 di6 ;
41 ." run " BOOT
42 run
43
```

Modyfikujemy słowo *alarm* [21]. Zamiast czasów dla timerów wstawiamy odpowiednio stałe *t1* i *t2* (1.0->t1; 3.0->t2). Teraz możemy zdefiniować słowo *czas_t1* [16], które otwiera okno parametryzacji dla podania nowej wartości czasu. Po wstawieniu nowej wartości i zatwierdzeniu jej OK zostanie wykonane słowo *def_t1*, które przedefiniowuje nam wartość stałej *t1*. Definicje te muszą być poprzedzone definicją stałej *t1*, jako wartości początkowej [12]. Analogicznie dla czasu *t2*. Na koniec definiujemy słowa *3menu* i *4menu* [26-27], które w kolejnych wierszach funkcji dodatkowych pozwalają wywołać okna parametryzacji *czas_t1* i *czas_t2*. Słowa *3menu* i *4menu* wywołujemy w słowie *run* [40].

2.8 Powiadomienie SMS

Modyfikujemy nasz program tak, aby przy zachodzącym warunku zostało wysłane powiadomienie SMS na telefon użytkownika. Naszym warunkiem będzie 10 cykl alarmu. To znaczy, że słowo alarm będzie mogło być wywołane nie więcej niż 10 razy i wtedy będzie automatycznie zatrzymane i zostanie wysłany SMS z treścią, jaką ustawimy za pomocą okna parametryzacji wywoływanego w menu funkcji dodatkowych. Budujemy blok programowy *komunikat SMS* [27]. Nasz program wygląda następująco:

```
1  ( MÓJ PIERWSZY PROGRAM
2  ( Nazwa projektu: MAX
3  ( data otwarcia projektu: 2010.09.17
4  ( Wersja: v1.0
5
6  STOPALL CLEARSYS
7  2 ERRORLEVEL
8  FORGET program
9  0 ERRORLEVEL
10  : program ." MAX v1.0 2010.09.17 " ;
11 ( definicja stałych t1 i t2
12 1.0 FCONSTANT t1
13 3.0 FCONSTANT t2
14 : def_t1 TOF t1 ;
15 : def_t2 TOF t2 ;
16 : czas_t1 ." t1 [sek] " t1 GET def_t1 STOP ;
17 : czas_t2 ." t2 [sek] " t2 GET def_t2 STOP ;
18 ( licznik cykli ALARM
19 0 1 VAR!
20 : plus 1 VAR? 1 + 1 VAR! ;
21 ( sterowanie wyjściem DO1
22 : on  1 1 DO! 0.5 3000 BEEP ;
23 : off 0 1 DO! 0.5 1000 BEEP ;
24 : alarm plus on t1 1 TIMER! off t2 2 TIMER! alarm ;
25 : stop 0.0 1 TIMER! STOP 0.0 2 TIMER! STOP 0 1 DO! 0 1 VAR! ;
26
27 ( komunikat SMS
28 ." KOMUNIKAT " 1 STRING!
29 ." +48123456789 " 1 USERPHONE
30 : string1 ." TEKST " 1 GETS STOP STOP ;
31 : sms 1 USER 1 STRING? SMS DROP ;
32 : limit 1 VAR? 10 = IF sms stop THEN 0.1 4 TIMER! limit ;
33
34 ( definicja funkcji w menu Funkcji dodatkowych
35 : 1menu ." ALARM " 1 MENU alarm ;
```

```
36 : 2menu ." STOP " 2 MENU stop ;
37 : 3menu ." Czas t1> " 3 MENU czas_t1 ;
38 : 4menu ." Czas t2> " 4 MENU czas_t2 ;
39 : 5menu ." TEKST SMS " 5 MENU string1 ;
40
41 ( definicja funkcji przycisków F1 i F2
42 : f1 F1 BUTTON alarm ;
43 : f2 F2 BUTTON stop ;
44
45 ( Wej. DI1 i DI2
46 0 5 FLAG!
47 0 6 FLAG!
48 : di5 5 DI? 5 FLAG? NOT AND IF alarm THEN 5 DI? 5 FLAG! 0.1 5 TIMER! di5 ;
49 : di6 6 DI? NOT 6 FLAG? AND IF stop  THEN 6 DI? 6 FLAG! 0.1 6 TIMER! di6 ;
50
51 ( słowo uruchamiające aplikację: run
52 : run 1menu 2menu 3menu 4menu 5menu f1 f2 di5 di6 limit ;
53 ." run " BOOT
54 run
55
```

Warunek, przy spełnieniu którego zostanie wysłane powiadomienie SMS definiujemy w słowie *limit* [32]. W słowie tym odczytujemy wartość zmiennej 1 i porównujemy ja do liczby 10. Za pomocą operatora warunkowego rozpatrujemy warunek i jeżeli wynik porównania jest prawdą (10=10) zostanie wykonane słowo *stop* i *sms* [31]. Słowo sms wysyła powiadomienie zapisane pod 1 zmienną wierszową (STRING) na numer telefonu zapisany pod 1 użytkownikiem (USER). Wartości początkowe tych parametrów musimy wstępnie zdefiniować [28-29]. Numer telefonu użytkownika nr 1 możemy też zdefiniować lub przedefiniować w menu konfiguracyjnym (MENU -> Użytkownicy -> TELEFONY -> TEL 1>). Do zmiany domyślnego tekstu powiadomienia za pomocą okna parametryzacji służy zdefiniowane słowo string1 [30], które wywoływane jest w poprzez słowo *menu5* [39] jako kolejna funkcja menu funkcji dodatkowych. Jeszcze tylko musimy stworzyć funkcje, która dokonuje inkrementacji zmiennej 1 wraz z kolejnymi cyklami słowa *alarm*. Tworzymy słowo *plus* [19], które będzie wykonywane w słowie *alarm*. W słowie run musimy wywołać słowa *5menu* i *limit*.

2.9 Okno drukowania i formatowanie tekstu.

Stworzymy blok programowy, który jest przykładem wyprowadzania danych na ekran sterownika oraz swobodnego formatowania tekstu. Całość formatowanego tekstu zamyka się w jednym słowie *print*. Będzie ono cyklicznie wywoływane przez słowo *print_cykl*. Odświeżanie ekranu będzie, co 1sek.

```
: print
  ." Moja pierwsza " NEWLINE ." aplikacja " GREEN 0 0 PRINT
  ." Stan: " BLACK 0 2 PRINT
  1 FLAG? IF ." ALARM " ELSE ." STOP " THEN RED 6 2 PRINT
  ." Temp:     " BLACK 0 3 PRINT NOAUTOSPACE 4 AI? F.
  ." stC " BLACK 6 3 PRINT AUTOSPACE
  ." Cykl:         " BLACK 0 4 PRINT 1 VAR? . RED 6 4 PRINT ;
: print_cykl print 1.0 3 TIMER! print_cykl ;

: run CLEAR 1 FPREC! 1menu 2menu 3menu 4menu 5menu f1 f2 di5 di6 limit print_cykl ;
```

W pierwszym i drugim wierszu będzie drukowany napis „Moja pierwsza aplikacja”. Tekst zawiera więcej niż 15 znaków i ma wyznaczony tylko jeden punkt startowy drukowania (0 0), ale poprzez „złamanie” go słowem NEWLINE rozłożymy go na dwie linie. W trzecim wierszu drukujemy słowo „Stan:” oraz w zależności od tego, czy alarm działa czy nie będzie drukowane słowo „ALARM” lub „STOP”. Zależy to od zmiennej bitowej 1 (1 FLAG!). Jeżeli jest 1, będzie drukowane „ALARM”, jeżeli 0 to „STOP”. Teraz trzeba jeszcze definiować odpowiednio te flagę w odpowiednim miejscu. W słowie *alarm* definiujemy flagę jako 1, a w słowie *stop* jako 0. Dodatkowo tekst w jednej linii jest drukowany w dwóch kolorach: czarnym i czerwonym.

```
: alarm 1 1 FLAG! on plus t1 1 TIMER! off t2 2 TIMER! alarm ;
25 : stop 0.0 1 TIMER! STOP 0.0 2 TIMER! STOP 0 1 DO! 0 1 VAR! 0 1 FLAG! ;
```

W czwartej linii drukujemy zmienną wartość 4 wejścia analogowego. Dla tego przykładu wejście to możemy ustawić jako

prądowe. Ustawiając precyzje drukowania 1, czyli z jedną liczbą po przecinku, otrzymamy wartość z zakresu 0,0÷21,3. Nie wymnażamy tej wartości przez określony współczynnik, tak jak to się robi dla uzyskania wartości rzeczywistej z zakresu czujnika pomiarowego ACP. Z początku wiersza drukujemy słowo „Temp:" z 8 spacjami. Służy to „przykryciu" końcówki tekstu, który może pozostać na końcu linii z poprzedniego cyklu drukowania, ponieważ naprzemiennie mogą być drukowane liczby jedno- i dwucyfrowe- części całkowitej. W celu łącznego drukowania wartości liczbowej z jednostką (°C) przed zestawieniem tekstu w buforze wejściowym posłużyliśmy się słowem NOAUTOSPACE. Po wydrukowaniu powracamy do automatycznego wstawiania spacji za pomocą słowa AUTOSPACE.
W piątym wierszu przywołując wartość zmiennej 1 (1 VAR?) będziemy drukować liczbę wykonanych cykli słowa *alarm*.
W słowie *run* definiujemy precyzje drukowania liczb jako 1 oraz dokładamy słowo CLEAR (czyszczące ekran przy restarcie sterownika) i *print_cykl* (uruchamiające w cyklu nasza funkcję drukowania *print*).

2.10 Status pracy – zapytanie i odpowiedź.

Umiemy już formatować tekst i drukować na ekranie. W taki sam sposób formatujemy tekst dla słów SMS oraz LOG. Możemy tez stworzyć słowo, które tylko wprowadza tekst do bufora wejściowego i nic więcej. Za pomocą takiego słowa możemy otrzymywać informacje o systemie, stanie wejść/wyjść lub parametrach współpracujących ze sterownikiem urządzeń.
Definiujemy słowo:

```
: status
." Stan: " 1 FLAG? IF ." ALARM " ELSE ." STOP " THEN NEWLINE
." Temp: " 4 AI? F. ." stC " ;
```

To słowo tworzy treść w buforze i nic więcej. A wiemy, że bufor wyjściowy jest drukowany na terminalu lub na wyświetlaczu telefonu przy wykonaniu słowa, które taki bufor tworzy. Więc wystarczy teraz podać słowo *status* w treści SMSa wejściowego, a tekst zostanie dopisany do automatycznej odpowiedzi. Dlatego nie musimy pisać słowa NAK, a otrzymamy tylko jednego SMSa na numer, z którego wysłano zapytanie stan. Nie musimy podawać żadnych numerów telefonów. Tekst będzie w tym przypadku taki:
Stan: ALARM
Temp: 18.7 stC

Nasza aplikacja przyjmuje ostateczny kształt:

```
1  ( MÓJ PIERWSZY PROGRAM
2  ( Nazwa projektu: MAX
3  ( data otwarcia projektu: 2010.09.17
4  ( Wersja: v1.0
5
6  STOPALL CLEARSYS
7  2 ERRORLEVEL
8  FORGET program
9  0 ERRORLEVEL
10  : program ." MAX v1.0 2010.09.17 " ;
11 ( definicja stałych t1 i t2
12 1.0 FCONSTANT t1
13 3.0 FCONSTANT t2
14 : def_t1 TOF t1 ;
15 : def_t2 TOF t2 ;
16 : czas_t1 ." t1 [sek] " t1 GET def_t1 STOP ;
17 : czas_t2 ." t2 [sek] " t2 GET def_t2 STOP ;
18 ( licznik cykli ALARM
19 0 1 VAR!
20 : plus 1 VAR? 1 + 1 VAR! ;
21 ( sterowanie wyjściem DO1
22 : on  1 1 DO! 0.5 3000 BEEP ;
23 : off 0 1 DO! 0.5 1000 BEEP ;
24 : alarm 1 1 FLAG! plus on t1 1 TIMER! off t2 2 TIMER! alarm ;
25 : stop 0.0 1 TIMER! STOP 0.0 2 TIMER! STOP 0 1 DO! 0 1 VAR! 0 1 FLAG! ;
26
27 ( komunikat SMS
28 ." KOMUNIKAT " 1 STRING!
```

```
29 ." +48123456789 " 1 USERPHONE
30 : string1 ." TEKST " 1 GETS STOP STOP ;
31 : sms 1 USER 1 STRING? SMS DROP ;
32 : limit 1 VAR? 10 = IF sms stop THEN 0.1 4 TIMER! limit ;
33
34 ( definicja funkcji w menu Funkcji dodatkowych
35 : 1menu ." ALARM " 1 MENU alarm ;
36 : 2menu ." STOP " 2 MENU stop ;
37 : 3menu ." Czas t1> " 3 MENU czas_t1 ;
38 : 4menu ." Czas t2> " 4 MENU czas_t2 ;
39 : 5menu ." TEKST SMS " 5 MENU string1 ;
40
41 ( definicja funkcji przycisków F1 i F2
42 : f1 F1 BUTTON alarm ;
43 : f2 F2 BUTTON stop ;
44
45 ( Wej. DI1 i DI2
46 0 5 FLAG!
47 0 6 FLAG!
48 : di5 5 DI? 5 FLAG? NOT AND IF alarm THEN 5 DI? 5 FLAG! 0.1 5 TIMER! di5 ;
49 : di6 6 DI? NOT 6 FLAG? AND IF stop  THEN 6 DI? 6 FLAG! 0.1 6 TIMER! di6 ;
50
51 : print
52 ." Moja pierwsza " NEWLINE ." aplikacja " GREEN 0 0 PRINT
53 ." Stan: " BLACK 0 2 PRINT
54 1 FLAG? IF ." ALARM " ELSE ." STOP " THEN RED 6 2 PRINT
55 ." Temp:      " BLACK 0 3 PRINT NOAUTOSPACE 4 AI? F.
56 ." stC " BLACK 6 3 PRINT AUTOSPACE
57 ." Cykl:          " BLACK 0 4 PRINT 1 VAR? . RED 6 4 PRINT ;
58 : print_cykl print 1.0 3 TIMER! print_cykl ;
59
60 ( SMS: status pracy
61 : status
62 ." Stan: " 1 FLAG? IF ." ALARM " ELSE ." STOP " THEN NEWLINE
63 ." Temp: " 4 AI? F. ." stC " ;
64
65 ( słowo uruchamiające aplikację: run
66 : run CLEAR 1 FPREC! 1menu 2menu 3menu 4menu 5menu f1 f2 di5 di6 limit print_cykl ;
67 ." run " BOOT
68 run
69
```