Wyświetlacze graficzne DWIN (1)

Sposób na szybkie wykonanie interfejsu dotykowego

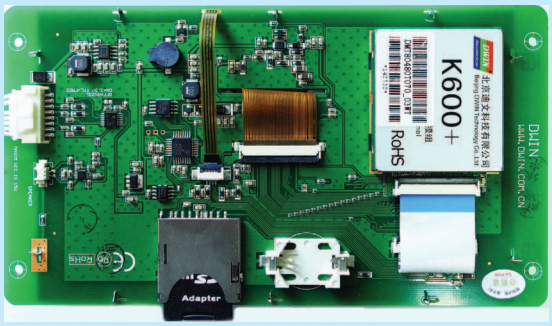
Redakcja Elektroniki Praktycznej dziękuje panu Krzysztofowi Bielińskiemu z firmy White Electronics (*www.whiteelectronics.pl*) za udostępnienie wyświetlacza do testów

Graficzne interfejsy użytkownika są niezbędnym elementem wielu urządzeń przemysłowych i codziennego użytku. Z punkty widzenia programisty czy konstruktora, obsługa wyświetlaczy graficznych, a szczególnie interakcja pomiędzy elementami grafiki na ekranie (na przykład przyciskami) jest skomplikowana programowo. Dlatego, aby w rozsądnym czasie i przy zachowaniu umiarkowanych kosztów można było wykorzystać możliwości wyświetlacza i/lub panelu dotykowego, stosuje się gotowe, płatne lub darmowe biblioteki funkcji graficznych. Jednak praktyczne wykorzystanie API zawartego w bibliotekach wymaga pewnej wiedzy, ale też i przede wszystkim – poświęcenia czasu na programowanie. W wielu projektach czas opracowania urządzenia jest krytyczny. Są też projekty, w których trzeba szybko i często zmieniać wygląd i funkcje interfejsu. Do takich produktów, chociaż nie tylko, przydadzą się wyroby firmy DWIN. Są to wyświetlacze graficzne z wbudowanym zaawansowanym układem sterowania oraz z programowymi narzędziami do tworzenia interfejsów graficznych.

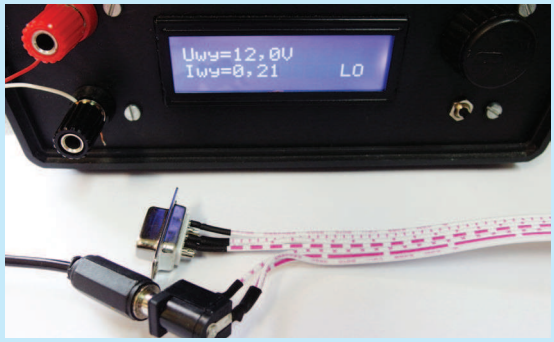
Część sprzętowa systemu składa się z matrycy LCD (TFT) zintegrowanej z zaawansowanym sprzętowym sterownikiem nazwanym (od nazwy rdzenia) K600+. W ofercie firmy jest wiele takich modułów różniących się wielkością i rozdzielczością matrycy, rodzajem zasilania i przeznaczeniem: dla aplikacji konsumenckich, przemysłowych, multimedialnych itp. Są też moduły przeznaczone do współpracy z systemem Android, wyposażone w gniazda USB, moduł Ethernetu i WiFi. Na **fotografii 1** pokazano płytkę wyświetlacza DMT 80480T070 o rozdzielczości 800×480 pikseli. Widać tu moduł K600+, gniazdo karty SD, gniazdo baterii litowej do podtrzymywania zegara RTC oraz układy interfejsu RS232/UART (tryb pracy jest wybierany przełącznikiem) i elementy stabilizatora impulsowego.

Projektowanie i programowanie interfejsów użytkownika odbywa się z wykorzystaniem firmowego pakietu IDE o nazwie DGUS_SDK (DWIN Graphics User System). Ponieważ wszystkie obliczenia wymagane przez funkcje graficzne wykonuje sterownik K600+,

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 63172, pass: 428ofq53



Fotografia 1. Płytka modułu DMT 80480T070 o rozdzielczości 800×480 pikseli



Fotografia 2. Kabel do przyłączenia zasilania i interfejsu RS232

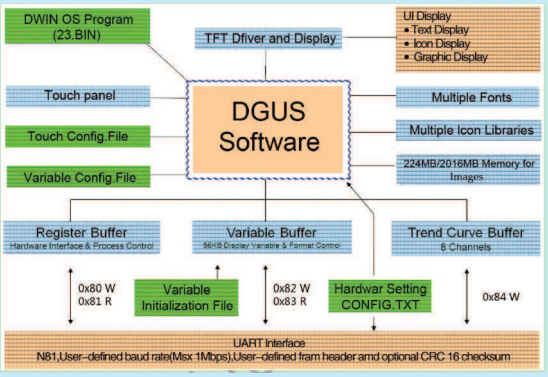
to komunikacja z hostem może być ograniczona do wymiany niewielkiej ilości danych. Moim zdaniem, najwygodniejszym interfejsem do przesyłania niewielkiej ilości danych jest UART. Niemal każdy, nawet bardzo prosty mikrokontroler ma wbudowany jeden lub więcej interfejsów UART, a jego obsługa jest nieskomplikowana. Co ważne, obsługa UART jest wspierana przez standardowe biblioteki dostarczane z każdym kompilatorem C.

Modułu DMT 80480T070 – pierwsze kroki

Do pierwszych testów systemu DWIN wykorzystałem moduł DMT 80480T070 o rozdzielczości 800×480 pikseli, pokazany na fot. 1. W komplecie z modułem dostarczono kabel HDL662, który jest z jednej strony zakończony 8-pinowym wtykiem włączanym do gniazda na płytce modułu, a z drugiej strony standardowym, żeńskim złączem DSUB9 (RS232) i współosiowym gniazdem zasilającym (**fotografia 2**).

Do gniazda zasilającego trzeba dołączyć źródło napięcia stałego z zakresu 5…15 V i obciążalności ok. 250 mA. Ja zasiliłem moduł wyświetlacza z zasilacza prądu stałego +12 V. Pobór prądu z trakcie pracy wynosił 210…220 mA. Tak zasilony moduł można dołączyć standardowym kablem do komputera PC wyposażonego w port RS232 lub w przejściówkę USB/RS232.

Wszystkie elementy interfejsu: bitmapy, zmienne i pliki konfiguracyjne są generowane przez DGUS-SDK. Użytkownik musi sobie te pliki przekopiować na kartę SD, a następnie umieścić tę kartę w złączu modułu. Istnieje możliwość bezpośredniego przesłania plików konfiguracyjnych przez interfejs RS232 lub w innych wariantach modułu – za pomocą USB.



Rysunek 3. Struktura programowania w systemie DGUS

Pakiet DGUS-SDK

Na **rysunku 3** pokazano strukturę oprogramowania systemu DGUS. Centralnym elementem jest program DGUS-SDK, w którym jest wykonywany projekt interfejsu dla K600+. Projekt zawiera pliki: sterujące, bitmap ekranów, ikon, znaków alfanumerycznych (fontów), z listą zmiennych itp. W DGUS SDK jest tworzony kompletny interfejs graficzny z definicjami interakcji pomiędzy interfejsem a użytkownikiem (obsługa ekranu dotykowego). Formalny opis programu można znaleźć w dokumentacji firmowej. Dobrym sposobem poznania tego narzędzia będzie pokazanie, jak powstaje projekt interfejsu, od bardzo prostego, z podstawowymi elementami, do bardziej rozbudowanego i skomplikowanego.

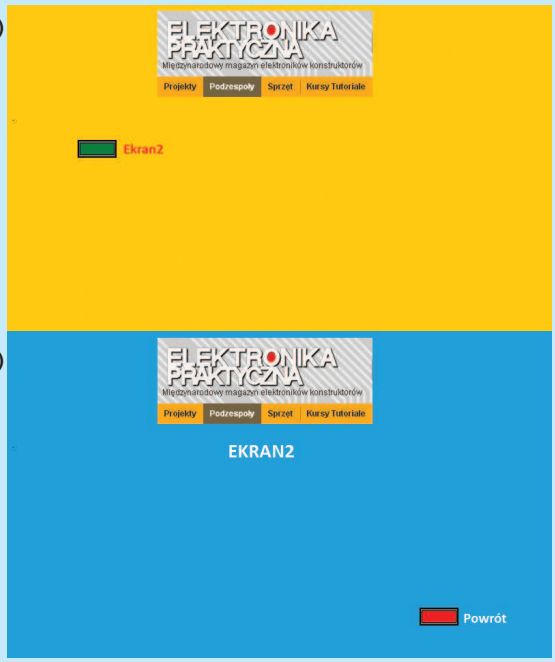
W momencie pisania artykułu było dostępna najnowsza wersja V5.0 beta DGUS – SDK różniąca się wyglądem interfejsu od bardziej znanej V4.9. Ta nowsza wersja była przeze mnie używana i zostanie opisana.

Po zainstalowaniu i uruchomieniu pakietu otwiera się zakładka *Welcome*, z której można:

- Otworzyć nowy projekt (New Project).
- Otworzyć wcześniej zapisany projekt (Open Project).
- Uruchomić jeden z czterech programów narzędziowych: edytor fontów, program konwersji plików, generator ikon i aplikacje do przesyłania danych przez UART.

Zaczynamy od kliknięcia na ikonę *New Project*. Pojawi się okno, w którym musimy wybrać: rozdzielczość ekranu z rozwijanego menu i ścieżkę dostępu do katalogu, w którym zostanie zapisany projekt i wszystkie jego pliki. Określenie prawidłowej rozdzielczości jest bardzo istotne. Wybranie innej rozdzielczości niż ma moduł, dla którego jest przeznaczony projekt spowoduje, że po wgraniu do modułu nie będzie on działał. W naszym wypadku wybieramy rozdzielczość 800×480 pikseli i przechodzimy do ekranu głównego. Ekran główny można podzielić na: obszar paska narzędzi, okno IMAGES zarządzania bitmapami ekranów, okno edycji ekranu i okno właściwości *Property*.

Idea tworzenia interfejsów graficznych opiera się na ekranach. Ekran to bitmapa o rozdzielczości równej rozdzielczości ekranu wyświetlacza. Oprogramowanie DGUS – SDK nie ma edytorów ekranów, trzeba je sobie przygotować w jakimś programie graficznym. Pomimo że dopuszczalne są bitmapy kompresowane w formatach JPG lub PNG, to najlepiej jest stosować nieskompresowane 24-bitowe bitmapy zapisywane jako pliki z rozszerzeniem „.BMP”. Są też określone zasady nazywania plików. Nazwa bitmapy ekranu głównego musi zaczynać się od zera. Może to być na przykład „*0.bmp*” lub

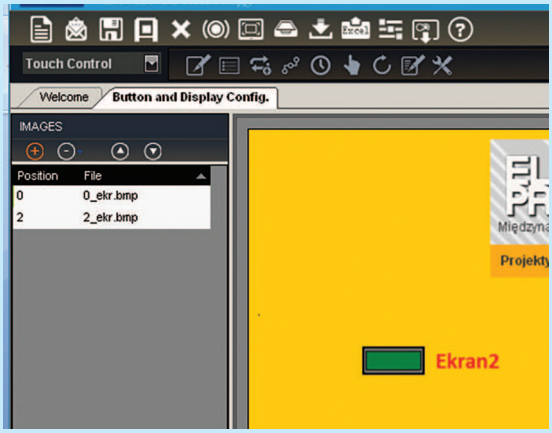


Rysunek 4. Ekrany projektu: a) ekran główny, b) ekran numer drugi

„0_ekran_glowny.bmp”, lub podobnie. Nazwy kolejnych plików bitmap muszą zaczynać się od kolejnych liczb, na przykład *„1_ekran_menu.bmp”*, *„2_ekran_podstrony.bmp”*, *„3_ustaw_zegar.bmp”* itd.

Dla opisywanego wyświetlacza trzeba przygotować bitmapę o rozdzielczości `800×480 pikseli. Jak wspomniałem, program nie ma edytora widżetów (obiektów) graficznych. Dlatego w trakcie przygotowywania bitmapy, jeśli chcemy mieć możliwość, na przykład, sterowania przyciskiem, to trzeba jego symbol umieścić na bitmapie ekranu. Ma to swoje zalety, ale ma też i wady. Brak edytora widżetów powoduje, że użytkownik musi symbole graficzne przygotować samodzielnie. Na początku wymaga to sporo pracy, bo trzeba sobie skompletować coś w rodzaju biblioteki elementów graficznych. Zaletą jest możliwość użycia dowolnego elementu przeznaczonego do sterowania – nie ma ograniczeń wbudowanej biblioteki. Przy włożeniu pewnego wysiłku można zaprojektować naprawdę ładne i funkcjonalne ekrany. Pokażę teraz jak w praktyce wygląda zaprojektowanie ekranu głównego z umieszczonym zielonym przyciskiem przechodzenia do kolejnego ekranu nazwanym *„Ekran 2”*. Na tym kolejnym ekranie jest umieszczony czerwony przycisk powrotu do ekranu głównego *„Powrót”*. Ten projekt pokazuje bardzo użyteczny mechanizm przełączania ekranów i sposób definiowania obsługi ekranu dotykowego, chociaż nie wywołuje żadnych funkcji poza graficznymi.

W pierwszym kroku zaprojektujemy dwa ekrany. Na każdym z nich jest umieszczony prostokątny przycisk, co pokazano na **rysunku 4**. W trakcie tworzenia nowego projektu, w jego katalogu jest tworzony katalog z plikami wynikowymi nazwany *DWIN_SET*. Pliki z wcześniej przygotowanymi bitmapami ekranów trzeba skopiować do tego katalogu. Teraz w oknie IMAGES klikamy na ikonkę „czerwony plus” i otwiera się okno wyboru plików bitmap. Jako pierwszy wybieramy plik *„0_ekr.bmp”*, a potem *„2_ekr.bmp”*. Dodane pliki pojawiają się na liście nawigacji plikami okna IMAGES . Po kliknięciu nazwy pliku z okna IMAGES jest on otwierany w oknie edycji (**rysunek 5**).



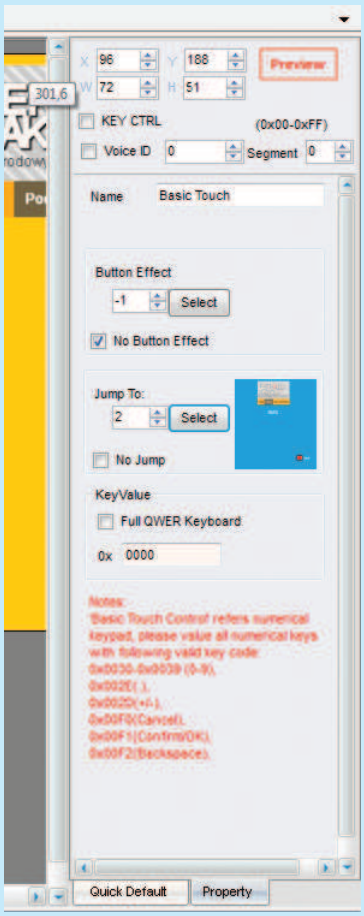
Rysunek 5. Dodanie dwóch ekranów

Teraz przychodzi moment, w którym obszarom, które chcemy by miały własności przycisku przydzielić możliwość współdziałania z ekranem dotykowym. Na pasku narzędzi, w lewym górnym rogu okna programu, wybieramy opcję *Touch Control* i następnie klikamy na ikonkę *Basic Touch Control*. W tym momencie możemy na otwartym do edycji ekranie definiować obszar prostokątny wykrywany przez driver obsługi ekranu dotykowego. Obszar powinien obejmować cały przycisk i nie może nakładać się na inne obszary, jeżeli zostały wcześniej zdefiniowane. Pokazano to na **rysunku 6**.

Kolejnym krokiem po określeniu obszaru jest zdefiniowanie akcji, która będzie wykonywana po naciśnięciu zaznaczonego obszaru. Z prawej strony otwieramy okno *Property* (**rysunek 7**). Teraz będą nas interesowały dwie ramki: *Jump To* i *Button Effect*. Najpierw zajmiemy się ramką *Jump To*. Definiuje się tutaj, gdzie zostanie wykonany skok po naciśnięciu przycisku lub inaczej mówiąc, która bitmapa ekranu zostanie wyświetlona. Po kliknięciu

Rysunek 6. Definiowanie obszaru działania ekranu dotykowego

na przycisk *Select* otwiera się okno z wszystkimi bitmapami dodanymi do projektu. Zaznaczamy bitmapę „*2_ekr.bmp*” i klikamy na „*OK*”. Teraz, po kliknięciu na obszar zielonego przycisku, powinien się otworzyć ekran drugi. Takie same czynności wykonujemy dla czerwonego przycisku ekranu drugiego: otwieramy bitmapę „*2_ekr.bmp*”, definiujemy obszar obsługiwany przez ekran dotykowy na czerwonym przycisku, w ramce *Jump To* wybieramy bitmapę „*0_ekr.bmp*”. Po przyciśnięciu zielonego przycisku na ekranie głównym przechodzimy do ekranu drugiego. Powrót z ekranu drugiego do ekranu głównego następuje po naciśnięciu czerwonego przycisku.



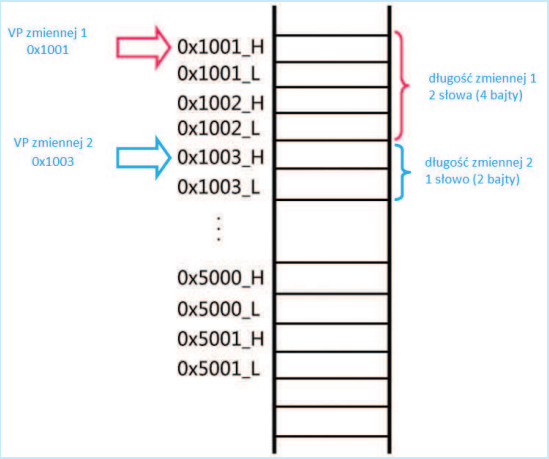
Rysunek 7. Okno *Property* polecenia *Touch Control*

Popatrzmy teraz jak działa druga ramka *Button Effect*. Jeżeli chcemy, aby w trakcie przyciskania przycisku zmieniał się jego kolor lub kształt, to musimy zdefiniować kolejną bitmapę „*1_ekr.bmp*” ze zmodyfikowanym przyciskiem umieszczonym w tym samym miejscu na ekranie. Ja zmieniłem kolor przycisku na biały i dodałem napis *PRESS*. Teraz w ramce *Button Effect* klikamy na *Select* i wybieramy bitmapę „*1_ekr.bmp*”. Po takiej definicji, w momencie przyciskania przycisku zielonego na ekranie głównym zmieni się jego wygląd, tak jak „*1_ekr.bmp*”.

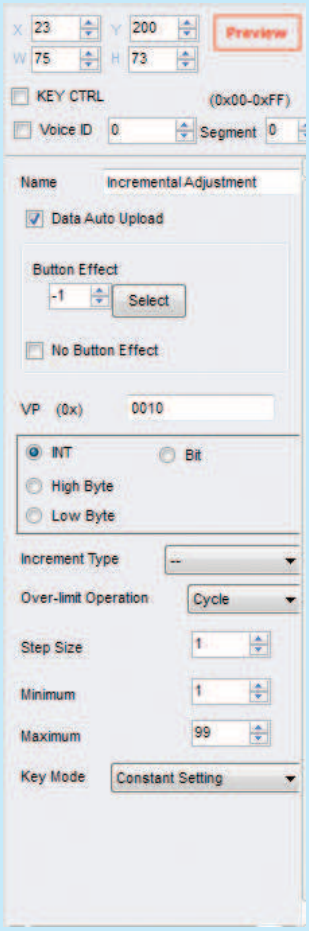
Zmiana ustawianej wartości

Zamiana jakiegoś parametru jest prawie zawsze wykonywana za pomocą interfejsu użytkownika. Przykładem może być ustawienie temperatur progowych i histerezy termostatu. Aby zmieniać parametr musi on być umieszczony w obszarze pamięci RAM, tak aby można go było odczytywać, modyfikować i zapisywać. W tym momencie warto powiedzieć parę słów o zmiennych w systemie DGUS.

Wszystkie zmienne używane w interfejsie są pamiętane w pamięci zmiennych (*Variable SRAM*) o wielkości 56 kB. Pamięć jest zorganizowana w słowa 2–bajtowe adresowane wskaźnikiem VP (*Variable Pointer*). Każda zmienna musi mieć swój unikalny VP nadany jawnie przez użytkownika, inaczej niż na przykład w kompilatorach języka C, gdzie deklarowane zmienne są umieszczane w pamięci przez kompilator a użytkownik poza wyjątkowymi przypadkami nie musi znać ich adresu. Zarządzanie zmiennymi i ich umieszczanie w pamięci, czyli przydzielanie adresu początku VP w pamięci w systemie DGUS jest zadaniem programisty tworzącego projekt. Żeby wykonać to prawidłowo, trzeba znać liczbę zmiennych i ich wielkość. Oprócz najmniejszych zmiennych 2–bajtowych można definiować i używać dłuższych zmiennych o mieszczących wielokrotność dwóch bajtów.



Rysunek 8. Adresowanie zmiennych



Rysunek 9. Okno *Property Incremental Adjustment*

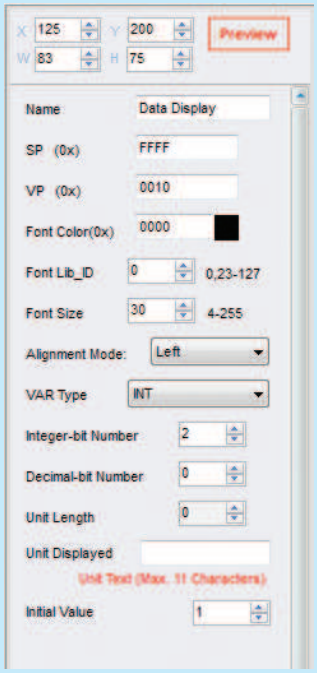
Na **rysunku 8** pokazano przykład adresowania dwóch zmiennych. Pierwsza ma VP=1001 i długość 2 słów. Przy tej długości druga zmienna może zaczynać się od adresu 1003 lub większego. Przy większej liczbie zmiennych może być potrzebna dokładna mapa przestrzeni adresowej wszystkich zmiennych z adresami VP i ich długością. Nadpisywanie zmiennych przez inne prowadzi do nieprawidłowego działania interfejsu. O zmiennych będzie jeszcze mowa przy okazji omawiania sposobu wymiany danych pomiędzy wyświetlaczem, a systemem mikroprocesorowym, do którego jest dołączony wyświetlacz realizujący funkcję interfejsu użytkownika.

Wróćmy teraz do zadania modyfikowania wartości zmiennej. Wykorzystamy do tego celu dwa przyciski: jeden zmniejszający, a drugi zwiększający jej wartość. Zgodnie z filozofią systemu najpierw przygotowujemy bitmapę z dwoma przyciskami. Ja zastosowałem dwie ikonki z symbolami „minus” i „plus”. Każdej z tych ikonek trzeba przypisać możliwość wykonywania akcji w powiązaniu z panelem dotykowym. Z paska narzędzi *Touch Panel* wybieramy element *Incremental Adjustment* i zaznaczamy na obu ikonach obszar aktywny dla drivera panelu dotykowego. Najpierw klikamy na obszar ikonki „minus” i otwieramy okno *Property* (**rysunek 9**).

Pierwszą rzeczą, którą musimy zrobić jest ustalenie adresu VP modyfikowanej zmiennej. Ja ustaliłem VP=0010. Ponieważ ten obszar dotyczy przycisku do zmniejszania wartości, to *Increment type* wybieramy „--”. Kolejne parametry oznaczają:

- *Step Size* – krok zmiany parametru ustawiamy na 1.
- *Minimum* – minimalna wartość parametru ustawiamy na 1.
- *Maksimum* – maksymalna wartość parametru ustawiamy na 99.
- *Over Limit Operation* – czy po osiągnięciu wartości granicznych ustawianie ma się zatrzymać (*Stop*), czy ma być kontynuowane modulo (*Cycle*).

Jako ostatni parametr ustawiamy *Key Mode*. Po wybraniu opcji *Constant Settings* przyciśnięcie i przytrzymanie przycisku powoduje cykliczną zmianę modyfikowanej zmiennej. Prawie dokładnie tak samo ustawiamy parametry okna „plus”. Jedyną różnicą jest *Increment type* – tu wybieramy „++”. Oczywiście VP również musi być równe 0010, bo będziemy zwiększać wartość tej samej zmiennej.



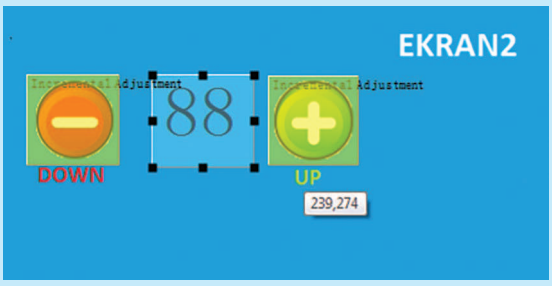
Rysunek 10. Okno *Property Data Display*

W kolejnym kroku musimy zdefiniować obszar z wyświetlaną wartością zmiennej. Zmieniamy pasek narzędzi z *Touch Control* na *Display Config* i wybieramy *Data Display*. Po ustaleniu obszaru wyświetlania otwieramy okno *Property* (**rysunek 10**) i kolejno ustawiamy:

- Font Lib_ID=0
- VP zmiennej – w naszym przypadku będzie to adres 0010.
- Parametry wyświetlanych znaków –kolor, wielkość czcionki i wyrównanie (lewy, prawy, środek).
- Rozmiar zmiennej (od 2 do 8 bajtów) i ilość cyfr części całkowitej i ilość cyfr części ułamkowej wyświetlanych po przecinku.
- Wyświetlane miano np. V, A itp.
- Wartość, z którą zmienna jest inicjowana.

Fragment bitmapy z zaznaczonymi definicjami obszarów *Data Display* i *Incremental Adjustment* pokazano na **rysunku 11**.

Po wygenerowaniu plików konfiguracyjnych, kopiujemy folder DWIN_SET na kartę SD i wkładamy do wyświetlacza. Program zostaje skopiowany do pamięci wewnętrznej i uruchomiony, pozwalając nam na sprawdze-

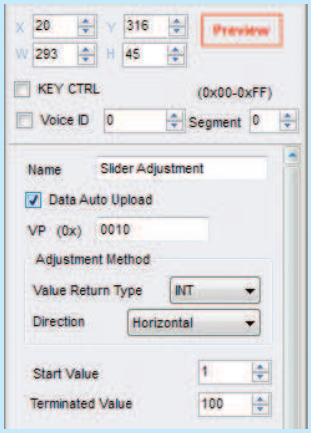


Rysunek 11. Fragment bitmapy z zaznaczonymi definicjami obszarów *Data Display i Incremental Adjustment*

nie jego działania. Po przytrzymaniu jednego z przycisków wartość jest modyfikowana automatycznie.

Suwak – slider

Suwak jest kolejnym elementem graficznym pozwalającym ustawiać wartości zmiennych. W kolejnym przykładzie pokażę jak go zdefiniować i z niego korzystać, ale w tym celu jest konieczne napisanie podstawowych informacji na temat definiowania ikon.



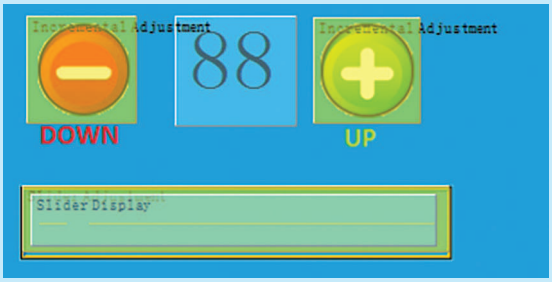
Rysunek 12. Okno *Property Slider Adjustment*

Ikony to elementy graficzne o dodatkowych możliwościach. W odróżnieniu od zwykłych bitmap mogą się przesuwać po ekranie. W przypadku slidera ikona będzie potrzebna do wizualizacji położenia przesuwającego się na ekranie suwaka w trakcie modyfikacji zmiennej. Do definiowania ikony jest przeznaczony program narzędziowy *DWIN Icon File*. Generator ikon uruchamiany z zakładki *Welcome DGUS–SDK*. Przed jego uruchomieniem trzeba narysować element graficzny w postaci bitmapy o odpowiednich wymiarach i kształcie. Nasz suwak będzie prostokątem o wymiarach 15×40 pikseli, liniach brzegowych w kolorze żółtym i wypełnieniu w kolorze czerwonym.

Po otwarciu programu *Icon File Generator* klikamy na przycisk *Select The Path of Image File* i wybieramy przygotowany plik lub, jeżeli jest taka potrzeba, więcej niż jeden plik. Potem klikamy na przycisk *Create the New Icon File*. Plik ikon ma rozszerzenie „.ICO" i jego nazwa musi się zaczynać od liczby z zakresu 23…127. Jeżeli ten wymóg nie będzie spełniony, to system nie odczyta pliku jako definicji ikon. Poza tym plik ikon musi być umieszczony w katalogu DWIN_SET, który jest potem kopiowany na kartę. Nasza ikona została zapisana w pliku *„32.ICO"*.

Po zdefiniowaniu ikony suwaka możemy przystąpić do wykonania slidera. W naszym przykładzie umieściłem go na ekranie drugim pod przyciskami zwiększania i zmniejszania wartości zmiennej o adresie VP=0010. Ponieważ podobny element jest zdefiniowanym widżetem w używanej przeze mnie wtyczce GDDX Microchipa, to z przyzwyczajenia narysowałem bardzo podobny element.

Żeby można było dotykowo sterować pracą suwaka z paska narzędzi *Touch Control* wybieramy *Slider Adjustment* i wyznaczamy obszar pokrywający rysunek i wykrywany przez driver ekranu dotykowego.

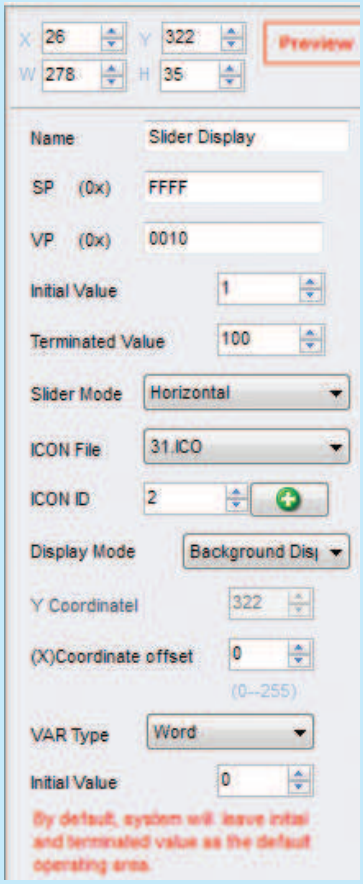


Rysunek 13. Fragment ekranu z elementami *Incremental Adjustment* oraz *Slider*

Przesuwanie po tym obszarze powoduje modyfikowanie zmiennej o adresie VP ustawianym w polu VP okna *Property Slider Adjustment* (**rysunek 12**). Celowo ustawiłem VP=0010, by suwak zmieniał wartość zmiennej już modyfikowanej i wyświetlanej przyciskami „plus" i „minus" z poprzedniego przykładu. Zobaczymy przez to ciekawą właściwość slidera pozwalającą mu pracować niejako w odwrotną stronę, to znaczy jako liniowy wskaźnik ustawianej wielkości. Poza VP ustawiamy jeszcze położenie poziome (Horizontal), wartość początkową (*Start Value*) i wartość końcową (*Terminated Value*). Teraz można by było już używać tego elementu, ale musimy jeszcze dodać graficzny wskaźnik położenia suwaka. Do tego celu wykorzystamy wcześniej narysowaną ikonę suwaka i element *Slider Display* z paska narzędzi *Display Control*. Na rysunku slidera za pomocą elementu *Slider Display* definiujemy obszar poruszania się ikony po rysunku lidera. W naszym wypadku obszary *Slider Adjustment* i *Slider Display* prawie się pokrywają. Zostało to pokazane na **rysunku 13**.

Ostatnim krokiem będzie skonfigurowanie właściwości Slider Display w oknie Property. Powiązanie za pomocą adresu VP ze zmienną pozwala na ustalenie położenia suwaka na tle rysunku slidera. Jeżeli *VP Slider Display* jest równe *VP Slider Adjustment*, to przesunięcie po ekranie dotykowym powoduje zmianę wartości zmiennej, a ta zmiana powoduje przesunięcie ikony suwaka.

Okno *Property Slider Display* pokazano na **rysunku 14**. Oprócz adresu VP i wartości początkowej i końcowej wybieramy plik z ikoną (*„31.ICO"*). W pliku może być więcej niż jedna ikona. Wyboru właściwej dokonujemy przez określenie jej ID. Jest to tym łatwiejsze, że po kliknięciu na symbol zielonego plusa wyświetla się okno ze wszystkimi ikonami i możliwością wyboru jednej w wybranym pliku ICO. Dodatkowo, ustawiamy sposób wyświetlania suwaka (*background* – widoczny, *transparent* – przeźroczysty) oraz jego położenie – poziome (*horizontal*) lub pionowe (*vertical*). Po wygenerowaniu plików konfiguracyjnych można sprawdzić działanie. Przesuwanie placem po ekranie powoduje zmianę wyświetlanej wartości zmiennej o adresie 0010 i przesuwanie się suwaka. Ponieważ klawiszami „plus" i „minus" również zmieniamy wartość zmiennej o adresie VP=0010, to automatycznie zmienia się też położenie suwaka. Jest to wspominana możliwość „analogowego" wskaźnika modyfikowanej wartości zmiennej. Rysunek slidera można na przykład wykonać w postaci skali, a suwak w postaci wskazówki.



Rysunek 14. Okno *Property Slider Display*

Tomasz Jabłoński, EP