Wyświetlacze graficzne DWIN (2)

Klawiatura alfanumeryczna

W poprzedniej części opisałem, jak można nawigować pomiędzy ekranami interfejsu oraz zmieniać wartości zmiennych przy użyciu elementów Incremental Adjustment i Slider, jednak jednym z najwygodniejszych sposobów wprowadzania danych jest użycie wirtualnej klawiatury. Dla danych liczbowych odpowiednia będzie klawiatura cyfrowa zawierająca klawisze numeryczne. Klawiatura alfanumeryczna jest używana do wprowadzania tekstu. System DGUS pozwala na bardzo elastyczne konfigurowanie klawiatury. Można zdefiniować tylko klawisze numeryczne, wybrane klawisze alfanumeryczne lub pełną klawiaturę alfanumeryczną.

Wyświetlacze graficzne DWIN (2) W pierwszym kroku trzeba narysować klawiaturę z takim rozmieszczeniem klawiszy, z jakiego będziemy korzystali w projektowanej aplikacji. Ja skorzystałem z jednego dostępnych w Internecie rysunków obrazujących rozmieszczenie klawiszy na klawiaturze komputera PC (**rysunek 15**).

Do programowej realizacji klawiatury będą potrzebne:

- *Text Input* – element wybierany z paska narzędzi *Touch Control*.
- *Text Display* – element wybierany z paska narzędzi *Display Control*.
- *Basic Touch Control* – element wybierany z paska narzędzi *Touch Control*.

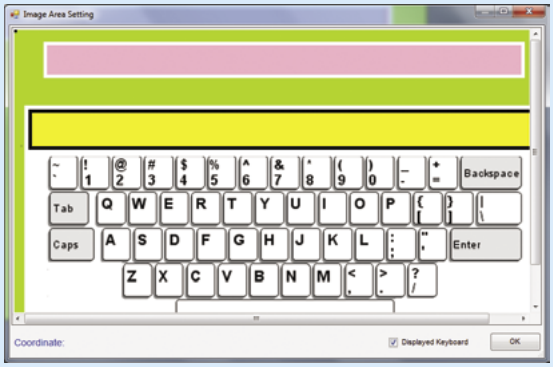
Element Text Input jest wykorzystywany do konfigurowania klawiatury i jej uruchamiania. Ponieważ jest to element z paska narzędzi *Touch Control*, to trzeba go umieścić na elemencie graficznym, na przykład ma przycisku. Ten przycisk może być opisany „Wprowadź tekst" lub podobnie. W naszym przykładowym projekcie, na stronie głównej umieściłem kolejny przycisk „Klawiatura". Po jego przyciśnięciu przechodzimy na kolejny ekran „3_ekr.bmp" z umieszczonym na nim przyciskiem „Wprowadź tekst", z naniesionym obszarem obiektu *Text Input*. W oknie właściwości (*Property*) tego elementu jest ustawianych większość parametrów pracy klawiatury. Najważniejsze z nich to:

Rysunek 15. Rysunek rozmieszczenia klawiszy wykorzystany w tym przykładzie

- *Jump To* – skok do ekranu z rysunkiem klawiatury. W naszym przypadku ta opcja jest wyłączona.
- *VP* – adres początku danych wprowadzanych przez klawiaturę.
- *Text length* – maksymalna długość wprowadzonego tekstu.
- *Text Display Area* – obszar na ekranie przeznaczony do wyświetlania wprowadzanych znaków. Można podać cyfrowe współrzędne lub zaznaczyć obszar na ekranie po kliknięciu na przycisk *Set*.
- *Keyboard Setting* – ustawienie ekranu z wyświetlana klawiaturą.
- *Font Lib_ID* – domyślnie we wszystkich elementach, w których występuje parametr *Font Lib_ID* jest ustawiona wartość 23 i wtedy wyświetlanie znaków nie działa. Trzeba pamiętać, aby tam wpisać zero.

Keyboard setting jest przeznaczony do definiowania obszaru z rysunkiem klawiatury, który będzie „wklejany" do ekranu, w którym będziemy wprowadzali tekst. Ten mechanizm jest bardzo użyteczny i efektowny. Można tak zdefiniować *Text Input*, że w momencie jego przyciśnięcia na ekranie pojawi się klawiatura. Po wprowadzeniu tekstu i przyciśnięciu OK klawiatura znika, a wprowadzony tekst jest wyświetlany przez element *Text Display*. Ponadto, na ekranie, z którego wywołaliśmy *Text Input* pozostają wszystkie graficzne elementy, które w trakcie wywołania klawiatury były przez nią przykryte.

Jak już wspomniałem, do definiowania wklejania rysunku klawiatury przeznaczony jest obszar *Keyboard Setting* z okna *Property* elementu *Text Input*. Po kliknięciu na niego pojawia się okno wyboru bitmapy z rysunkiem klawiatury. Tu będzie to pokazany na rys. 15 ekran zapisany w pliku „4_ekr.bmp". Po wybraniu tego ekranu pojawia się następne okno, w którym zaznaczamy część ekranu wklejaną jako rysunek klawiatury. Oczywiście, zaznaczamy całą naszą klawiaturę i zatwierdzamy przyciskiem OK. Po zdefiniowaniu obszaru klawiatury pozostaje teraz pokazać programowi, gdzie go ma wkleić na ekranie, na którym jest umieszczony element *Text Input* – w naszym wypadku jest to ekran „3_ekr.bmp".



Rysunek 16. Definiowanie obszaru wprowadzanego tekstu

Klikamy na przycisk *Set Paste Position* i na ekranie zaznaczamy położenie wklejanej klawiatury.

Wróćmy jeszcze na chwilę do *Text Display Area*. Po kliknięciu na przycisk Set pojawia się ekran „3_ekr.bmp" i na nim zaznaczamy, gdzie będzie się wyświetlał wprowadzany tekst. Dużym ułatwieniem jest opcja *Displayed Keyboard*. Po jej zaznaczeniu na ekranie pojawi się wklejona klawiatura i można precyzyjnie ustalić obszar wyświetlania wprowadzanego tekstu (**rysunek 16**). Ja do rysunku klawiatury dodałem żółty prostokąt i zaznaczyłem go jako obszar wprowadzania tekstu.

Po kliknięciu na klawisz OK naszej klawiatury powinna całkowicie zniknąć łącznie z wprowadzanym tekstem. Można dodać kolejny element *Text Display*, aby wprowadzony tekst był dalej widoczny.

W trakcie wpisywania znaków tekst jest wyświetlany w obszarze określonym przez *Text Display Area* elementu *Text Input* (tu jest to żółty prostokąt). Po kliknięciu na OK rysunek klawiatury znika , a wprowadzony tekst jest wyświetlany przez *Text Display* w obszarze otoczonym białą obwódką, umieszczonym na górze ekranu – widać go na rys. 16 . Powiązanie pomiędzy elementami *Text Input* i *Text Display* jest realizowane przez adres VP i parametr *Text Length*. Adres VP we właściwościach *Text Input* powinien być równy adresowi ustawianemu we właściwościach *Text Display*. Jest to adres początku bufora z wpisywanymi znakami ASCII. W jednym słowie są zapisane dwa znaki 8-bitowe. Dla parametru *Text length* =44 zajęte są 22 kolejne lokacje w pamięci zmiennych. Trzeba o tym pamiętać projektując mapę pamięci zmiennych.

Po zdefiniowaniu właściwości obiektów *Text Display* i *Text Input* trzeba przypisać każdemu klawiszowi możliwość generowania kodu. Do tego celu wykorzystamy obiekt *Basic Touch Control*. Na każdym z klawiszy rysunku klawiatury definiujemy obszar *Basic Touch* i we właściwościach przypisujemy kolejne kody klawiszy (**rysunek 17**). Jest to żmudna czynność, ale w jej wykonaniu pomaga okno wyboru kodu klawisza. W oknie *Property* zaznaczamy *Full QWER Keyboard* i klikamy na przycisk *Set*. Zostaje wyświetlone okno z możliwością wyboru klawiszy klawiatury, jak na **rysunku 18**. Po wyborze klawisza jego kod jest automatycznie przypisywany przyciśnięciu definiowanego obszaru *Basic Touch Control*. W przypadku klawiatury ASCII przypisujemy pełny kod klawisza zdefiniowany dla komputerów PC.

Żeby jednoznacznie zidentyfikować przyciśnięcie klawisza, można dodać kolejną bitmapę ekranu ze zmienionymi kolorami klawiszy. W trakcie definiowania Property Basic Touch Control trzeba w obszarze *Button*

Rysunek 17. Definicja obszarów klawiszy wirtualnych

Effect wybrać ten ekran. Wtedy przy każdym przyciśnięciu klawisza jego kolor będzie się zmieniał z białego na czerwony. Oczywiście, po puszczeniu kolor zmieni się ponownie na biały.

Zegar RTC – ustawianie

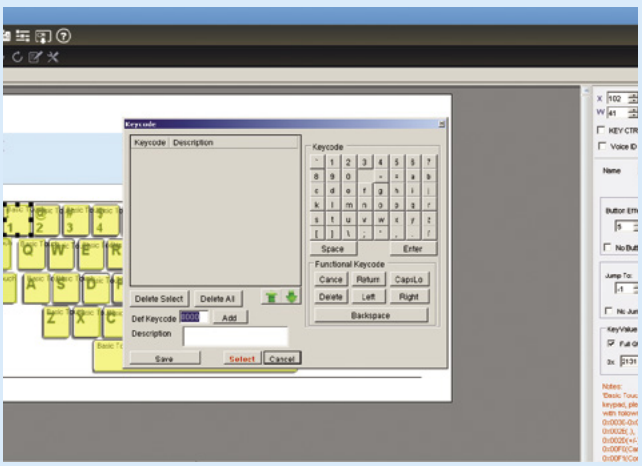
W moduł wyświetlacza jest wbudowany zegar czasu rzeczywistego z kalendarzem. Żeby przetestować jego działanie na ekranie głównym został dodany kolejny przycisk – RTC. Po jego przyciśnięciu przechodzimy do ekranu „6_rtc.bmp", na którym są umieszczone:

- Element *RTC* z paska narzędzi *Display Config* odpowiadający za wyświetlenie czasu w postaci cyfrowej.
- Przycisk *SET RTC* z elementem RTC z paska narzędzi *Touch Control* przeznaczony do ustawiania zegara.
- Przycisk *POWRÓT* z elementem *Touch Basic Control* z paska narzędzi *Touch Control* przeznaczony do powrotu do strony głównej.

Postawienie elementu RTC z paska narzędzi *Display Config* umożliwia wyświetlenie zegara z kalendarzem w postaci cyfrowej. W oknie *Property* można ustawić kolor czcionki, wielkość znaków i format wyświetlania czasu. Domyślnie jest ustawione kolejno: rok, miesiąc, dzień, godziny, minuty, sekundy i dzień tygodnia. Czas można również wyświetlać w postaci zegara analogowego animacja ruchu wskazówek wykorzystujący do tego celu ikony.

Używanie zegara wiąże się nieodłącznie z możliwością ustawiania czasu i daty. Żeby to zrobić trzeba użyć elementu *RTC* z paska narzędzi *Touch Control*. U nas ten element jest postawiony na przycisku *SET RTC*. W oknie *Property* ustawiamy:

- *Display Position* – miejsce, w którym będą wyświetlane ustawiane dane.
- *Font Color* – kolor czcionki.
- *Font Size* – wielkość czcionki.
- *Keyboard Setting* – konfigurowanie klawiatury.



Rysunek 18. Przypisanie kodu klawisza „1"

Konfigurowanie klawiatury odbywa się praktycznie tak samo, jak dla klawiatury alfanumerycznej: wybieramy ekran z klawiaturą, potem obszar wklejania i miejsce wklejania. Ja umieściłem klawiaturę na tym samym ekranie, więc efekt wklejania nie będzie widoczny. Klawiatura jest numeryczna. Oprócz klawiszy cyfr są tam umieszczone klawisze BCK (strzałka), ESC i OK. Każdemu klawiszowi trzeba przypisać obszar *Basic Touch Control* z kodem klawisza. Do tego celu wykorzystujemy 8-bitowe kody ASCII klawiszy. Kody należy wpisać ręcznie korzystając z informacji wypisanej czerwona czcionką pod oknem *Key Value*. Przypisanie 16-bitowych kodów klawiszy z komputera PC za pomocą *Full QWER Keyboard* powoduje, że w przypadku ustawiania zegara klawisze nie działają.

Ustawianie zegara rozpoczynamy od przyciśnięcia przycisku *Set RTC*. W obszarze zdefiniowanym przez *Display Position* okna *Property* elementu *RTC* (*Touch Control*) zostanie wyświetlona liczba 20 i szereg znaków zapytania. Przyciskając klawisze numeryczne wpisujemy kolejne ustawienia daty i czasu zgodnie z wybranym formatem. Zegar zostaje ustawiony po przyciśnięciu klawisza OK (kod 0xF1). Ekran z ustawianiem zegara został pokazany na **rysunku 20**.

Komunikacja z hostem

Prezentowany tutaj wyświetlacz jest z założenia jednym z elementów budowanego urządzenia. Jak to już powiedzieliśmy, jest to element realizujący graficzny interfejs użytkownika. Drugim elementem jest mikrokontroler – host wykonujący zasadnicze funkcje sterowania, regulacji, pomiarów itp. Zależnie od wykonywanych funkcji może to być prosty mikrokontroler lub zaawansowana jednostka 32-bitowa. Ponieważ urządzenie jest podzielone na dwa elementy niezależne układowo, to konieczna jest wymiana informacji pomiędzy nimi. Do komunikacji wybrano interfejs UART. Z pozycji wyświetlacza można sprzętowo wybrać poziomy napięć: według standardu RS232 lub +3,3 V dla stanu wysokiego i 0 V dla stanu niskiego.

Interfejs RS232 umożliwia przyłączenie wyświetlacza do komputera PC lub dowolnego innego wyposażonego w ten interfejs. Transmisja może się odbywać na odległości rzędu kilkunastu metrów (zależnie od wybranej prędkości transmisji i pojemności kabla połączeniowego). Standard +3,3 V można wykorzystać do zrealizowania połączenia na małe odległości – na przykład, gdy wyświetlacz i host są w tej samej obudowie. Upraszcza to interfejs, ponieważ nie potrzeba driverów RS232.

Parametry transmisji od strony wyświetlacza określa się w oknie *System Config* otwieranym z górnego paska narzędzi (**rysunek 21**). Dla transmisji szeregowej ustawiamy:

- Prędkość transmisji (baud rate). Można wybrać jedna ze standardowych prędkości. Jeżeli potrzebujemy niestandardowych prędkości, to można je zdefiniować zapisując rejestry R5 i R9 w oknie *User-defined Baud Rate*. Sposób obliczenia wartości tam zapisywanych można znaleźć w dokumentacji.
- Bajty nagłówka. Protokół wymiany wymaga wysyłania dwóch zdefiniowanych bajtów nagłówka. Zapisuje się je w polach rejestrów R3 i RA.
- W oknie FCRC można odblokować konieczność obliczania i przesyłania 2 bajtów wielomianu kontrolnego CRC. W aplikacjach wymagających dużej niezawodności działania pozwala to na wyeliminowanie błędnych (przekłamanych) ramek protokołu transmisji. Sposób obliczania wielomianu łącznie z tablica współczynników jest umieszczony w dokumentacji.

Rysunek 19. Klawiatura alfanumeryczna i wprowadzanie tekstu

Do celów testowych ustawiłem prędkość transmisji 57600 bps, bez obliczania CRC i bajty nagłówka równe 0xAA i 0x55.

Wymiana informacji pomiędzy wyświetlaczem, a hostem będzie polegała na dostępie (zapisywaniu i odczytywaniu zmiennych adresowanych przez wskaźnik VP w oknach *Property* elementów oraz na dostępie do rejestrów wyświetlacza. Najpierw zajmiemy się dostępem do zmiennych.

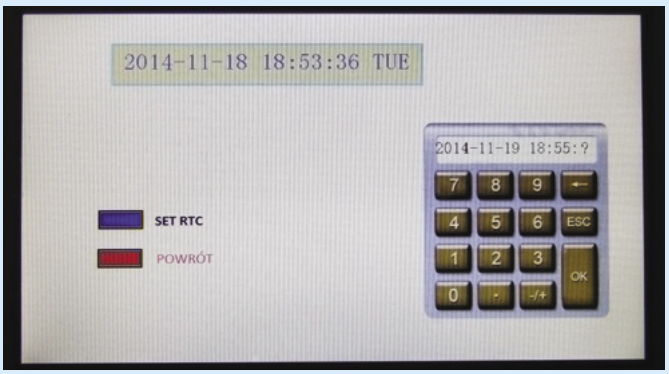
Protokół wymiany danych pomiędzy hostem a wyświetlaczem składa się z pięciu kolejnych pól:

- Pola nagłówka zawierającego opisywane wyżej dwa bajty definiowane w oknie konfiguracji.
- Pola długości danych. Jest to jeden bajt zawierający liczbę pozostałych danych w protokole – bez bajtów nagłówka i bajta pola długości danych.
- Pola komendy. Jest tu zapisywany jeden bajt z kodem komendy.
- Pola danych. Są tu mieszczone dane o ilości określonej w polu długości danych, jeżeli nie włączyliśmy przesyłania CRC lub o liczbie określonej w polu długości danych – 2 jeżeli włączyliśmy przesyłanie CRC.
- Opcjonalnego pola CRC zawierającego 2 bajty sumy kontrolnej CRC.

Wykaz komend umieszczono w **tabeli 1**.

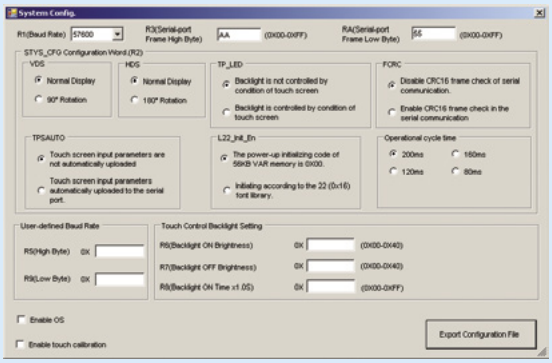
Żeby zapisać dane do zmiennej musimy znać jej adres, czyli wartość VP. Na **rysunku 22** pokazano ramkę służącą do zapisania zmiennej o adresie 0x0010. Była ona wykorzystywana do testowania elementów *Incremental Adjustment* i *Slider*

Po wysłaniu danych z rys. 23 do zmiennej o adresie 0x0010 zapisze się wartość 0x0032 (50dec). Żeby

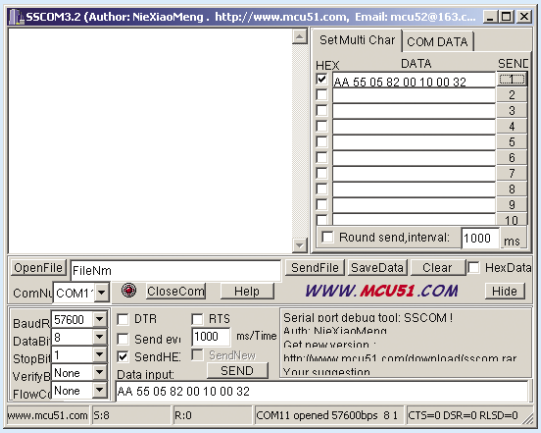


Rysunek 20. Ustawianie zegara RTC

Tabela 1. Wykaz komend			
Komenda	**Kod**	**Pozostałe dane**	**Opis**
Dostęp do rejestru	0x80	Adres rejestru + dane	Zapis zaadresowanego rejestru
	0x81	Adres Rejestru +ilość danych	Odczytanie danych z zaadresowanego rejestru
		Adres Rejestru +ilość danych+ dane	Odpowiedź z modułu DGUS
Dostęp do zmiennych w pamięci RAM	0x82	Adres rejestru + dane	Zapis zaadresowanego obszaru RAM
	0x83	Adres Rejestru +ilość danych	Odczytanie danych z zaadresowanego obszaru RAM
		Adres Rejestru +ilość danych+ dane	Odpowiedź z modułu DGUS
Zapis bufora kołowego	0x84	Bajt trybu + dane	Zapis bufora kołowego



Rysunek 21. Okno konfiguracji



Rysunek 23. Okno programu do testowania transmisji *Serial Debugging Assistant*

sprawdzić poprawność modyfikacji zmiennej można użyć do tego celu komputer PC z prostym oprogramowaniem pozwalającym na wysyłanie ciągu znaków przez interfejs RS232. Oczywiście, jest potrzebny komputer z portem RS232, a o taki coraz trudniej. Producent pomyślał i o tym dostarczając „przejściówkę" USB/RS232 oraz potrzebne drivery. Do testowania transmisji można użyć np. programu *Serial Debugging Assistant* (*sscom32.exe*) dostarczanego razem z programem DGUS SDK (**rysunek 23**).

Przed użyciem trzeba skonfigurować port RS232 podając prędkość transmisji, ilość bitów danych, ilość bitów stopu, bit parzystości i kontrolę przepływu danych. Po kliknięciu na przycisk EXT pojawi się okno, w którym można wpisywać znaki do wysłania. Wpisujemy tam ramkę danych AA 55 05 82 00 10 00 32 i klikamy na przycisk „1", umieszczony obok okna wprowadzania danych. Jeżeli port RS232 jest poprawnie skonfigurowany, to po wysłaniu ramki na ekranie 2 wyświetlana wartość powinna zmienić się na 50. Test można powtórzyć zmieniając wartość ostatniego bajta. W ten sposób można sprawdzić poprawność działania komendy zapisywania danej pod konkretny adres.

Bardzo podobnie sprawdzimy odczytywanie zawartości zmiennej. Odczytywanie przebiega w 2 etapach. W pierwszym etapie wysyłamy do wyświetlacza komendę z kodem 0x83, a potem odczytujemy z wyświetlacza żądaną ilość danych. Żeby odczytać jedno słowo zmiennej o adresie 0x0010 trzeba wysłać komendę pokazana na **rysunku 24**. Wyświetlacz po odebraniu komendy odsyła do hosta zawartość zmiennej – **rysunek 25**.

W ten sam sposób można wyczytać wprowadzany tekst przy pomocy klawiatury alfanumerycznej. Pamiętamy, że zdefiniowaliśmy adres VP=0x0100 i długość 44 znaki ASCII. W pamięci SRAM wyświetlacza zostaną zarezerwowane 22 słowa 2-bajtowe. Po wysłaniu komendy odczytu o kodzie 0x83, adresie 0x01 0x00 i liczbie słów do przeczytania równej 0x16 (22 dzies.) wyświetlacz odeśle ramkę pokazana na rys. 25.

Przy opisywaniu okna *Property* części obiektów z paska narzędzi *Touch Control* celowo pominąłem jedną z opcji – *Data Auto Upload*. Teraz, gdy zajmujemy się odczytywaniem zawartości zmiennych, to jest dobry moment, aby do tego wrócić.

Wiemy w tym momencie jak odczytywać wartość zmiennych. Host wysyła komendę odczytu – wyświetlacz odpowiada. Załóżmy, że chcemy wiedzieć czy wartość zmiennej modyfikowanej przez *Incremental Adjustment* się zmieniła, ale w momencie tej zmiany. Żeby to zrobić trzeba by było ciągle wysyłać komendy odczytywania zmiennej, czyli stosować metodę poolingu. Jeżeli mamy tych zmiennych zdefiniowanych więcej, sprawa się komplikuje tym bardziej, im więcej jest zmiennych. Oprogramowanie DGUS ma możliwość działania zdarzeniowego powiązanego z elementami *Touch Control*. Każdy element *Touch Control*, do którego jest przypisana zmienna adresowana przez VP, ma możliwość włączenia opcji *Data Auto Upload*. Jeżeli jest ona włączona, to w momencie zmiany tej zmiennej wyświetlacz automatycznie wysyła poprzez UART ramkę z danymi, tak jakby odpowiadał na komendę odczytania zmiennej. Jest to typowy przykład działania zdarzeniowego. Kiedy panel dotykowy wykryje przyciśnięcie i powiązaną z tym zmianę wartości zmiennej, to generuje zdarzenie w postaci wysłania przez złącze szeregowe ramki z danymi. Teraz z kolei host musi ciągle nasłuchiwać czy nie została wysłana do niego ramka z danymi. Jest to o tyle proste,

Nagłówek	Pole długości danych	Pole komendy	Adres SRAM	Dane
0xAA 0X55	0x05	0x82	0x00 0x10	0x00 0x32

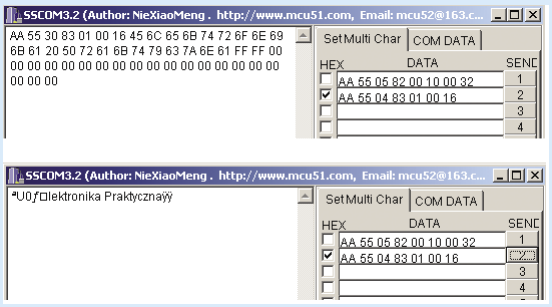
Rysunek 22. Ramka danych z komendą zapisania zmiennej o adresie 0x0010

Nagłówek	Pole długości danych	Pole komendy	Adres SRAM	Ilość czytanych danych
0xAA 0X55	0x04	0x83	0x00 0x10	0x01

Rysunek 24. Komenda odczytania zmiennej

Nagłówek	Pole długości danych	Pole komendy	Adres SRAM	Ilość danych	Dane
0xAA 0X55	0x06	0x83	0x00 0x10	0x01	0x00, 0x32

Rysunek 25. Dane wysyłane przez wyświetlacz do hosta



Rysunek 26. Odczytanie bufora ze znakami wprowadzanymi z klawiatury alfanumerycznej

że można do tego celu wykorzystać przerwania i proces nasłuchu pracuje w tle. Kiedy ramka zostanie skompletowana i przeanalizowana, to jest ustawiany znacznik i program główny może zająć się reakcją na odebraną ramkę.

Typowy element *Basic Touch* nie ma możliwości generowania zdarzenia, bo nie jest powiązany ze zmienną adresowaną przez VP. Jednak fakt przyciśnięcia przycisku jest bardzo często istotny dla algorytmów sterowania. Jeżeli jest taka konieczność, to można zastosować element *Return Key Code* wybierany z paska narzędzi *Touch Control*. Ten element może generować zdarzenie po przyciśnięciu, bo jest powiązany ze zmienną adresowaną VP, a poza tym można dla niego zdefiniować zwracany kod.

Przy aktywowaniu opcji *Data Auto Upload* trzeba pamiętać, że jest ona włączana globalnie w oknie *System Config*. Musi być wtedy zaznaczona opcja *TPSAUTO: Touch Screen Input Parameters automatically uploaded to the serial port*. Domyślnie ta opcja nie jest włączona i trzeba ją uaktywnić (**rysunek 26**).

Odczytywanie i zapisywanie rejestrów przebiega podobnie. W oprogramowaniu wyświetlacza jest zdefiniowanych szereg rejestrów sterujących jego pracą. Modyfikując zawartość rejestrów można uzyskać efekty nie możliwe do uzyskania innymi metodami. Za pomocą odczytywania i zapisywania rejestrów można miedzy innymi:

- Sterować jasnością podświetlania.
- Modyfikować czas piknięcia buzzera przy przyciskaniu klawiatury.
- Odczytywać ID wyświetlanego ekranu i zmieniać ID ekranu (skok do kolejnego ekranu).
- Odczytywać status panelu dotykowego: pierwsze przyciśnięcie, ciągłe przyciśnięcie, puszczenie.
- Odczytywać współrzędne przyciśnięcia klawisza.
- Blokować/ odblokować panel dotykowy.
- Odczytywać i ustawiać RTC.
- Wykonywać operacje na programowych timerach, programowo resetować układ wyświetlacza.

W naszym wypadku najłatwiej będzie pokazać, jak można ustawić przez hosta zegar RTC.

Z zegarem RTC są związane 2 rejestry:

- RTC_COM-ADJ pod adresem 0x1f – odblokowanie możliwości zapisu (ustawiania) RTC przez port RS232 i wysłanie nowej daty i czasu.
- RTC_NOW pod adresem 0x20 – rejestr z zapisanym czasem i datą; można go odczytywać.

Do zapisywania rejestrów jest przeznaczona komenda o kodzie 0x80, a odczytywania o kodzie 0x81 (tab. 1).

Najpierw wyślemy komendę służącą do odczytania zegara (**rysunek 27**). Czytamy 7 bajtów. Odczytane dane są zapisane znakami ASCII w kodzie BCD. Na przykład 18:33:12 będzie reprezentowana przez ciąg znaków ASCII „183312”.

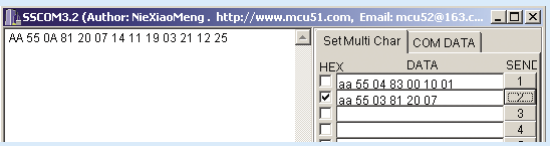
Format odpowiedzi jest następujący: rok (2 ostanie cyfry), miesiąc, dzień, dzień tygodnia, godziny, minuty, sekundy. Na **rysunku 28** pokazano odpowiedź na komendę odczytania dla daty: *2014-11-19 03(środa) 21:12:25*.

Do zmiany daty i czasu użyjemy komendy zapisy rejestru 0x1f pokazanej na **rysunku 29**.

Po przesłaniu adresu rejestru trzeba wysłać bajt 0x5a informujący wyświetlacz, że będzie ustawiany zegar, a następnie bajty daty i czasu według tego samego formatu, który jest przesyłany przy doczytywaniu rejestru o adresie 0x20.

Podsumowanie

Wyświetlacze firmy DWIN to bez wątpienia bardzo ciekawa propozycja dla tych, którzy chcieliby zastosować atrakcyjny i funkcjonalny graficzny interfejs użytkownika. Czy dla wszystkich? To oczywiście zależy od rodzaju projektowanego urządzenia, przewidywanej wielkości



Rysunek 28. Ramka z czasem odczytanym z rejestru 0x20

Nagłówek	Pole długości danych	Pole komendy	Adres Rejestru	Ilość czytanych danych
0xAA 0X55	0x03	0x81	0x20	0x07

Rysunek 27. Komenda odczytania zegara

Nagłówek	Pole długości danych	Pole komendy	Adres Rejestru	Zapis przez UART	Ustawiany czas : 2014-11-19 22:00:00 środa
0xAA 0X55	0x0A	0x81	0x1F	0x5A	0x14,0x11, 0x20, 0x04, 0x22, 0x00, 0x00

Rysunek 29. Komenda ustawiania zegara przez UART

serii, umiejętności programistów itp. Na pewno będzie to atrakcyjna oferta dla tych, którzy nie mają dostatecznych umiejętności, aby zaprogramować samodzielnie graficzny interfejs, ale też dla tych, dla których to programowanie jest zbyt drogie lub po prostu chcą oszczędzić czas przymierzając się do wytworzenia krótkiej serii lub indywidualnego zlecenia. Wykonanie interfejsu od początku do końca dla kilku sztuk urządzenia jest prawdopodobnie nieopłacalne, bo zajmuje dużo czasu doświadczonego programisty.

Firma DWIN oferuje dużo, by taki interfejs został wykonany szybko, nawet bardzo szybko nawet przez kogoś, kto się z tym wyświetlaczem spotyka pierwszy raz. To olbrzymia zaleta tego rozwiązania i trudno ją przecenić. Oczywiście nic nie jest za dramo i wyświetlacz z wbudowanym sterownikiem jest droższy, niż sam panel wyświetlacza bez wbudowanego kontrolera.. Korzystając z obszernej oferty producenta można dobrać wyświetlacz do potrzeb i uniknąć „przewymiarowania", czyli stosowania urządzenia o możliwościach, ale też o cenie wyższej niż potrzeba.

Opis wyświetlacza z konieczności został ograniczony do kilku przykładów. Możliwości systemu są oczywiście większe i trudno byłoby je wszystkie tu opisać. Przy pierwszym kontakcie z systemem odniosłem wrażenie, że całość jest trochę niedopracowana. Jednak im dłużej pracowałem z DGUS SDK, tym bardziej to wrażenie mijało. Okazało się, że mimo „siermiężnego" interfejsu w SDK wszystko jest logicznie i spójnie zaprojektowane oraz pomyślano o wielu możliwościach. Jedyna rzecz, która nie podobała mi się, to pliki pomocy. Brakuje tam czegoś w rodzaju „pierwsze kroki", gdzie pokazane byłoby krok po kroku tworzenie prostego przykładu z dokładnym opisaniem wszystkich funkcji i zależności w działaniu. Teoretycznie, niby jest coś takiego, ale przynajmniej dla mnie na początku zupełnie niezrozumiałe i potraktowane zbyt skrótowo. Są też pliki video, ale one również jakoś niezbyt mi przypadły do gustu. Dużo większą pomocą okazały się znalezione w sieci przykłady użytkowników i przykłady zamieszczone na stronie polskiego dystrybutora *whiteelectronics.pl*. Kiedy już się pozna podstawowe zasady działania, to bardziej zaawansowana pomoc jest dość przydatna.

Tomasz Jabłoński, EP