Obsługa pojemnościowych paneli dotykowych

Na przykładzie panelu zintegrowanego z wyświetlaczem Winstar WF35QTIBCDBC0#

W swojej praktyce projektowej wielokrotnie korzystałem z możliwości funkcjonalnych, które daje nam w projektowanym urządzeniu zintegrowanie nowoczesnego panelu dotykowego z wyświetlaczem. Dziś chyba już nikt nie ma wątpliwości, że elementy tego typu stały się standardowym i nieodzownym wyposażeniem wielu urządzeń powszechnego użytku, gdyż stawiają one na zupełnie nowej płaszczyźnie interakcję urządzenia z użytkownikiem czyniąc ją bardziej naturalną i... ludzką.

W tej chwili trudno sobie wyobrazić nowoczesny system nawigacji satelitarnej, współczesny telefon komórkowy czy też mały, przenośny komputer bez interfejsu dotykowego, a skoro taki (moim zdaniem słuszny) jest kierunek rozwoju współczesnej elektroniki użytkowej, czemu nie skorzystać z możliwości, jakie daje nam ten prosty, a zarazem genialny element. Tym bardziej, iż za sprawą dalekowschodnich producentów jest on w zasięgu możliwości finansowych większości, jak myślę, elektroników.

Jak wiadomo, pierwsze z elementów tego typu, które zastosowano w urządzeniach powszechnego użytku, bazowały na zasadzie pomiaru rezystancji dwóch przeźroczystych folii rezystancyjnych przedzielonych dielektrykiem, których to dotknięcie (a w zasadzie naciśnięcie) powodowało powstanie miejsca styku i odpowiednią zmianę stosunku rezystancji tak utworzonego dzielnika rezystancyjnego, który odpowiednio spolaryzowany pozwala na odczyt położenia miejsca styku tychże folii. Rozwiązanie tego typu, choć proste i tanie w produkcji, ma szereg wad. Panel rezystancyjny wymagał kalibracji, często musiał być obsługiwany za pomocą rysika, zaś same folie odznaczały się stosunkowo małą trwałością (wystarczy przypomnieć sobie jakość obsługi pierwszych telefonów komórkowych wyposażonych w tego rodzaju panel dotykowy lub też tanich systemów nawigacji satelitarnej, które z uwagi na cenę, nadal korzystają z tego typu rozwiązań). Jak to zwykle bywa, dość szybko poradzono sobie z tym problemem, wprowadzając do powszechnego użycia pojemnościowe panele dotykowe, które poza rozwiązaniem wielu problemów, z jakimi borykały się panele rezystancyjne, pozwoliły na wprowadzenie nowych, niedostępnych dotąd funkcjonalności, jak: obsługa wielu punktów jednocześnie, detekcja i obsługa tzw. gestów czy też wrażliwość na siłę nacisku. Poza tym, konstrukcja tego rodzaju peryferiów zapewniła im znacznie większą trwałość, jak i zdecydowanie zmniejszyła niekorzystny wpływ na jakość wyświetlanego obrazu na zintegrowanym zeń wyświetlaczem.

Ekran dotykowy wykonany w technologii pojemnościowej, co oczywiste, działa na innej zasadzie, aniżeli ekrany rezystancyjne. W paneluch pojemnościowych jest stosowanych szereg różnych rozwiązań, które różnią się konstrukcją i zasadą działania, lecz dla typowych ekranów o najbardziej popularnych rozmiarach nieprzekraczających zwykle 10”, jest stosowane rozwiązanie bazujące na wykorzystaniu zjawiska zmiany pola elektrycznego w miejscu dotknięcia szklanej płytki, na której napylono szereg niewidzialnych elektrod (równoległych do siebie linii), osobno dla osi X i Y. W przypadku dotknięcia ekranu tego typu, w miejscu tegoż dotknięcia zmienia się wartość pojemności elektrostatycznej na pobliskich elektrodach, w związku, z czym dość łatwo i precyzyjnie można określić miejsce dotyku. Dodatkowym atutem tego rodzaju rozwiązania jest możliwość detekcji wielu dotknięć w tym samym czasie (tzw. *multi touch*), duża szybkość reakcji, bardzo wysoka precyzja zapewniająca detekcję nawet delikatnych „muśnięć” ekranu, odporność detekcji na wodę i kurz, jak i (dzięki zastosowaniu zintegrowanego sterownika z odpowiednim oprogramowaniem) zdolność do detekcji różnych gestów użytkownika. Potencjalną wadą tego rodzaju rozwiązania jest fakt, że panele takie wymagają dotknięcia gołym palcem lub specjalnym rodzajem wskaźnika, są wrażliwe na znajdujące się w pobliżu metalowe elementy oraz fakt, że zwykle nie można z nich korzystać w rękawiczkach, choć tę ostatnią niedogodność wyeliminowano w ostatnich rozwiązaniach stosowanych w telefonach komórkowych.

Mając te, podstawowe informacje na temat naszego, ciekawego peryferium przejdźmy do rozwiązania praktycznego, którego bardzo dobrym przykładem jest pojemnościowy panel dotykowy ze zintegrowanym sterownikiem zastosowany w nowej serii kolorowych wyświetlaczy ciekłokrystalicznych TFT firmy Winstar, na przykład model o przekątnej ekranu 3,5” o oznaczeniu WF35QTIBCDBC0#, który charakteryzuje się następującymi właściwościami użytkowymi:

- Rozdzielczość 320×240 pikseli.
- Zintegrowany sterownik ekranu SSD1963 wyposażony w wiele unikalnych funkcji sprzęto-

wych ułatwiających rysowanie standardowych elementów ekranu.
- 16-bitowa głębia kolorów (w standardzie RGB565).
- 16-bitowa magistrala danych (w tym, konkretnym modelu).
- Zintegrowana przetwornica do podświetlenia ekranu.
- Zintegrowany sterownik pojemnościowego panelu dotykowego z rodziny FT5X06 firmy FocalTech Systems, Ltd. (co zresztą skrzętnie jest ukrywane przez producenta wyświetlacza TFT).

Jako, że obsługa wyświetlaczy TFT wyposażonych w sterownik SSD1963 była tematem mojego artykułu z numeru EP 8/2011, jak również rozwiązania praktycznego pod postacią systemu intelliDom (z EP 10/2011 i EP 11/2011) w tej chwili nie będę powracał do tej tema-

Tabela 1. Rozmieszczenie i opis wyprowadzeń 36-pinowego, standardowego złącza typu ZIFF wyświetlacza WF35QTIBCDBC0#

Pin	Symbol	Opis
1	GND	Masa zasilania
2	VDD	Napięcie zasilania (3.3V). Prąd zasilania ok. 150mA.
3	BL_E	Sterowanie podświetleniem ekranu (H: włączone, L:wyłączone)
4	RS	Sygnał wyboru rodzaju danych sterownika ekranu: Polecenie sterujące/Dana
5	WR	Sygnał żądania operacji zapisu sterownika ekranu
6	RD	Sygnał żądania operacji odczytu sterownika ekranu
7	DB0	16- bitowa magistrala danych sterownika ekranu
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	DB8	
16	DB9	
17	DB10	
18	DB11	
19	DB12	
20	DB13	
21	DB14	
22	DB15	
23	NC	Niewykorzystane
24	CTP_INT	Wyjście /INT kontrolera panelu dotykowego (sygnalizacja zdarzeń)
25	CS	Sygnał wyboru układu sterownika ekranu (aktywacji chipsetu)
26	RST	Sygnał Reset sterownika ekranu
27	NC	Niewykorzystane
28	NC	
29	CTP_SCL	Magistrala I^2C, sygnał zegarowy kontrolera panelu dotykowego
30	CTP_SDA	Magistrala I2C, sygnał danych kontrolera panelu dotykowego
31	CTP_RST	Sygnał Reset kontrolera panelu dotykowego
32	CTP_WAKE	Wejście usypiania/wybudzania kontrolera panelu dotykowego
33	VLED-	Masa zasilania podświetlenia panelu dotykowego (przetwornicy)
34	VLED-	
35	VLED+	Napięcie zasilania podświetlenia panelu dotykowego (przetwornicy)
36	VLED+	

Ważniejsze rejestry kontrolera panelu dotykowego FT5X06

Rejestr: DEVICE MODE (0x00)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne		M2	M1	M0				

Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa bieżący tryb pracy sterownika FT5X06: 0x00→Normalny tryb pracy sterownika (domyślnie), 0x40→Tryb testu.

Rejestr: GEST_ID (0x01)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne	G7	G6	G5	G4	G3	G2	G1	G0

Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa rodzaj wykrytego gestu użytkownika: 0x00→Brak gestu, 0x10→Przesunięcie w górę, 0x14→Przesunięcie w lewo, 0x18→Przesunięcie w dół, 0x1C→Przesunięcie w prawo, 0x48→Gest pomniejszenia, 0x49→Gest powiększenia.

Rejestr: TD_STATUS (0x02)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne					S3	S2	S1	S0

Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa liczbę wykrytych dotknięć ekranu (funkcja *multi touch*): 1…10.

Rejestr: TOUCH1_XH (0x03)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne	F1	F0			TX11	TX10	TX9	TX8

Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa rodzaj zdarzenia dla pierwszego dotyku (bity F1…F0) jak również 4 najstarsze bity współrzędnej X pierwszego miejsca dotyku (TX11…TX8). Dostępne są następujące wartości bitów flagi zdarzenia: 0x00→Naciśnięcie panelu, 0x01→Koniec naciśnięcia panelu, 0x02→Dalsze naciskanie panelu.

Rejestr: TOUCH1_XL (0x04)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0

Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa 8 najmłodszych bitów współrzędnej X pierwszego miejsca dotyku.

Rejestr: TOUCH1_YH (0x05)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne	ID3	ID2	ID1	ID0	TY11	TY10	TY9	TY8

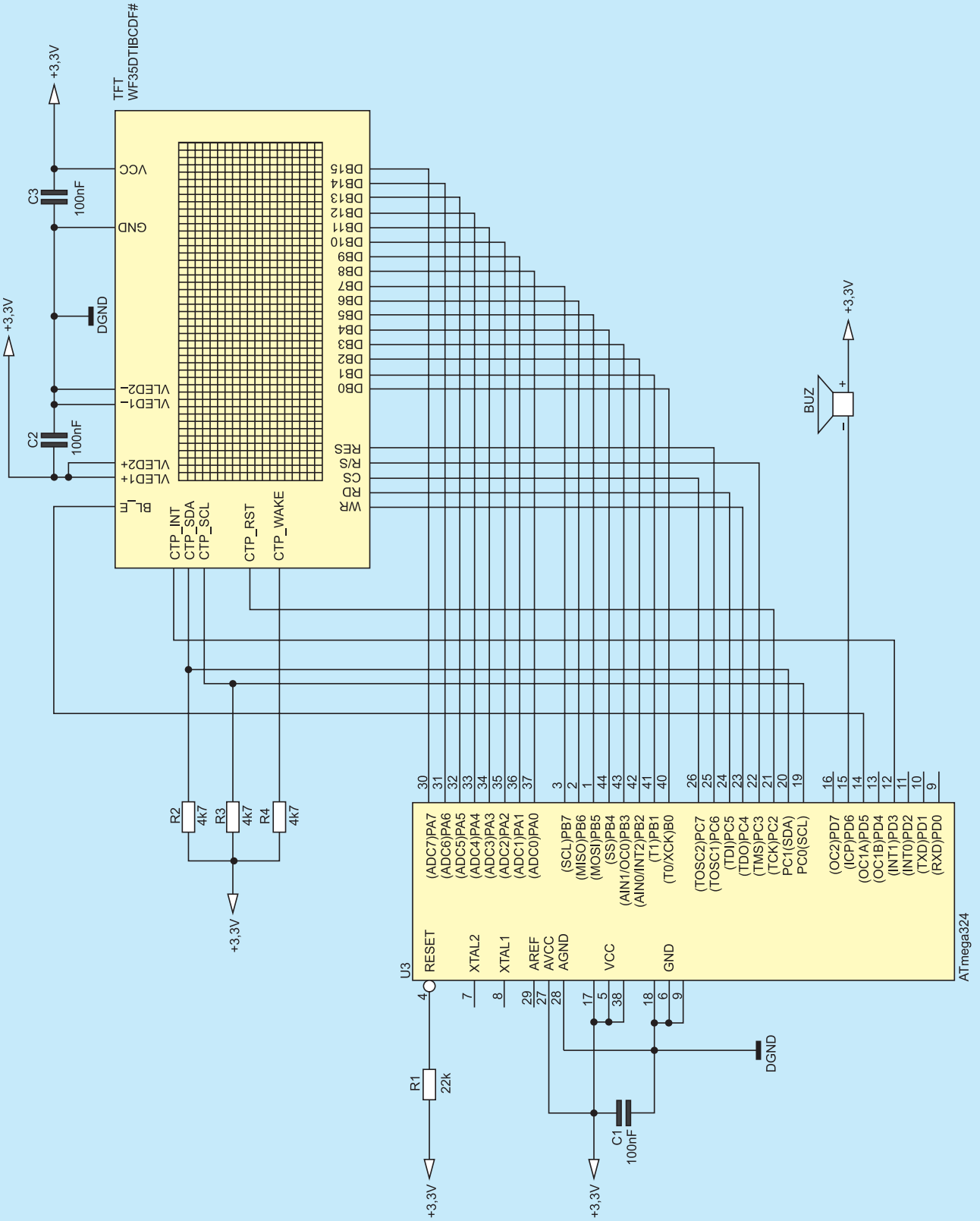
Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa identyfikator dotyku (bity ID3…ID0, niewykorzystywane) jak również 4 najstarsze bity współrzędnej Y pierwszego miejsca dotyku (TY11…TY8).

Rejestr: TOUCH1_YL (0x06)								
Bity	D7	D6	D5	D4	D3	D2	D1	D0
Aktywne	TY7	TY6	TY5	TY4	TY3	TY2	TY1	TY0

Wartość tego rejestru, przeznaczonego wyłącznie do odczytu, określa 8 najmłodszych bitów współrzędnej Y pierwszego miejsca dotyku.

tyki, lecz skupię się na samej obsłudze pojemnościowego panelu dotykowego. Wspomniany sterownik FT5X06 ma następujące właściwości użytkowe:

- Obsługa do 10 punktów dotyku w tym samym czasie (multi-touch).
- Sygnalizacja zdarzeń typu: dotknięcie panelu, dalszy kontakt, koniec dotknięcia panelu (zdjęcie palca z panelu).
- Sygnalizacja gestów: przesunięcie w górę, w dół, w prawo, w lewo, gest powiększenia, gest zmniejszenia.
- 3 tryby pracy zróżnicowane pod względem funkcjonalności i poboru energii: Active, Monitor i Hibernate.
- Wyjście /INT przeznaczone do sygnalizacji zdarzeń do kontrolera- hosta (aktywny stan niski).



Rysunek 1. Typowy schemat aplikacyjny wyświetlacza WF35QTIBCDBC0# z zastosowaniem mikrokontrolera ATmega324.

Tabela 2. Adresy i funkcje rejestrów

Nazwa rejestru	Adres rejestru	Funkcja
TOUCH2_XH	0x09	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X drugiego miejsca dotyku
TOUCH2_XL	0x0A	8 najmłodszych bitów współrzędnej X drugiego miejsca dotyku
TOUCH2_YH	0x0B	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y drugiego miejsca dotyku
TOUCH2_YL	0x0C	8 najmłodszych bitów współrzędnej Y drugiego miejsca dotyku
TOUCH3_XH	0x0F	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X trzeciego miejsca dotyku
TOUCH3_XL	0x10	8 najmłodszych bitów współrzędnej X trzeciego miejsca dotyku
TOUCH3_YH	0x11	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y trzeciego miejsca dotyku
TOUCH3_YL	0x12	8 najmłodszych bitów współrzędnej Y trzeciego miejsca dotyku
TOUCH4_XH	0x15	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X czwartego miejsca dotyku
TOUCH4_XL	0x16	8 najmłodszych bitów współrzędnej X czwartego miejsca dotyku
TOUCH4_YH	0x17	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y czwartego miejsca dotyku
TOUCH4_YL	0x18	8 najmłodszych bitów współrzędnej Y czwartego miejsca dotyku
TOUCH5_XH	0x1B	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X piątego miejsca dotyku
TOUCH5_XL	0x1C	8 najmłodszych bitów współrzędnej X piątego miejsca dotyku
TOUCH5_YH	0x1D	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y piątego miejsca dotyku
TOUCH5_YL	0x1E	8 najmłodszych bitów współrzędnej Y piątego miejsca dotyku
TOUCH6_XH	0x21	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X szóstego miejsca dotyku
TOUCH6_XL	0x22	8 najmłodszych bitów współrzędnej X szóstego miejsca dotyku
TOUCH6_YH	0x23	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y szóstego miejsca dotyku
TOUCH6_YL	0x24	8 najmłodszych bitów współrzędnej Y szóstego miejsca dotyku
TOUCH7_XH	0x27	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X siódmego miejsca dotyku
TOUCH7_XL	0x28	8 najmłodszych bitów współrzędnej X siódmego miejsca dotyku
TOUCH7_YH	0x29	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y siódmego miejsca dotyku
TOUCH7_YL	0x2A	8 najmłodszych bitów współrzędnej Y siódmego miejsca dotyku
TOUCH8_XH	0x2D	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X ósmego miejsca dotyku
TOUCH8_XL	0x2E	8 najmłodszych bitów współrzędnej X ósmego miejsca dotyku
TOUCH8_YH	0x2F	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y ósmego miejsca dotyku
TOUCH8_YL	0x30	8 najmłodszych bitów współrzędnej Y ósmego miejsca dotyku
TOUCH9_XH	0x33	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X dziewiątego miejsca dotyku
TOUCH9_XL	0x34	8 najmłodszych bitów współrzędnej X dziewiątego miejsca dotyku
TOUCH9_YH	0x35	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y dziewiątego miejsca dotyku
TOUCH9_YL	0x36	8 najmłodszych bitów współrzędnej Y dziewiątego miejsca dotyku
TOUCH10_XH	0x39	Rodzaj zdarzenia i 4 najstarsze bity współrzędnej X dziesiątego miejsca dotyku
TOUCH10_XL	0x3A	8 najmłodszych bitów współrzędnej X dziesiątego miejsca dotyku
TOUCH10_YH	0x3B	Identyfikator dotyku i 4 najstarsze bity współrzędnej Y dziesiątego miejsca dotyku
TOUCH10_YL	0x3C	8 najmłodszych bitów współrzędnej Y dziesiątego miejsca dotyku

Listing 1. Ciała funkcji odpowiedzialnych za obsługę interfejsu TWI mikrokontrolera ATmega32A

```
void TWI_Init(void)
{
    TWBR = 6;  //F_TWI=395kHz @ fosc = 11059200 Hz
}

void TWI_Start(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);
    while (!(TWCR&(1<<TWINT)));
}

void TWI_Stop(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
    while (TWCR&(1<<TWSTO));
}

void TWI_WriteByte(uint8_t Byte)
{
    TWDR = Byte;
    TWCR = (1<<TWINT)|(1<<TWEN);
    while ( !(TWCR&(1<<TWINT)));
}

uint8_t TWI_ReadByte(uint8_t ACK_NACK)
{
    TWCR = (1<<TWINT)|(ACK_NACK<<TWEA)|(1<<TWEN);
    while ( !(TWCR & (1<<TWINT)));
    return TWDR;
}
```

- Wejście /WAKE przeznaczone do usypiania/wybudzania sterownika panelu pojemnościowego (aktywny stan niski).
- Zintegrowany interfejs komunikacyjny I²C z trybem high-speed 400 kHz.
- Szereg sprzętowych rejestrów konfiguracyjnych pozwalających na szczegółową konfigurację parametrów sterownika związanych z detekcją dotyku i algorytmem sterującym dla różnych warunków pracy (zawilgocenie, krople wody, zabrudzenie).
- Napięcie zasilania z zakresu 2,8…3,6 V.
- Szeroki zakres temperatury pracy -40…+85°C.
- Obsługa przekątnych ekranu do 8,9" (aktualnie 3,5"; 4,3"; 5,7" i 7").
- Rozdzielczość 896×640 dla panelu o przekątnej 3,5", 1280×768 dla 4,3", 1408×1024 dla 5,7", 1792×1024 dla 7".
- Wbudowany mikrokontroler z rodziny 8051 z 28 kB pamięci Flash, 6 kB pamięci RAM i 256 B pamięci EEPROM.

W **tabeli 1** umieszczono rozkład wyprowadzeń 36-pinowego, standardowego złącza typu ZIFF (raster 0,5 mm),

w które wyposażono wyświetlacz WF35QTIBCDBC0#, jak i pozostałe, nowe modele firmy Winstar wraz z opisem wyprowadzeń.

Jak to zwykle bywa, obsługa urządzeń peryferyjnych tego typu polega na odczycie/zapisie specjalnych rejestrów stanu/konfiguracyjnych, dzięki którym możemy skonfigurować sterownik panelu oraz odczytywać bieżący stan urządzenia. Mimo że sterownik FT5X06 ma szereg możliwości w zakresie drobiazgowej konfiguracji algorytmu detekcji dotyku w różnych warunkach pracy panelu, i konfigurowania trybów pracy sterownika, standardowe, zdefiniowane przez producenta ustawienia są w zupełności wystarczające w zdecydowanej większości aplikacji, więc przedstawię wyłącznie te rejestry, za pomocą których jest możliwa ocena najważniejszych zdarzeń z punktu widzenia aplikacji użytkownika. Typowy schemat aplikacyjny wyświetlacza WF35QTIBCDBC0# z zastosowaniem mikrokontrolera ATmega324 w roli kontrolera-hosta, pokazano na **rysunku 1**.

Jak widać, jest to dość prosty system mikroprocesorowy, który do obsługi wbudowanego w wyświetlacz TFT sterownika panelu dotykowego wykorzystuje sprzętowy interfejs TWI mikrokontrolera (kompatybilny z interfejsem I²C) oraz przerwanie zewnętrzne INT1, które to wyzwalane jest za każdym razem, gdy sterownik FT5X06 wykryje obsługiwane zdarzenie nadzorowanego przezeń pojemnościowego panelu dotykowego. Przykładowe przebiegi na wyprowadzeniu /INT sterownika panelu dotykowego podczas naciskania obsługiwanego przezeń pojemnościowego panelu dotykowego pokazano na **rysunku 2**.

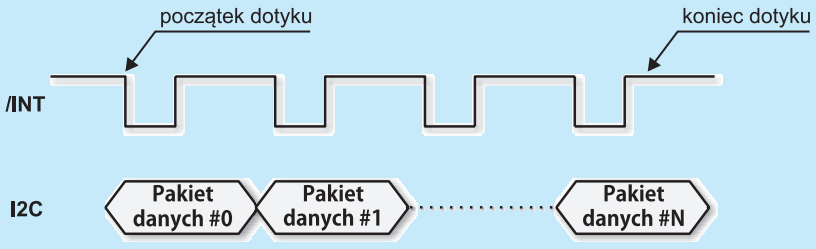
Dodatkowo, wprowadzono możliwość sygnalizacji dotknięcia panelu dotykowego przez prosty buzzer piezoelektryczny ze zintegrowanym generatorem lub też opcjonalnie (co wykorzystuje się dość często w aplikacjach telefonów komórkowych) za pomocą wibrującego silniczka o specjalnej konstrukcji. Mechanizm ten wykorzystuje wbudowany w strukturę mikrokontrolera Timer sprzętowy, dzięki któremu jest możliwe odmierzanie zadanego czasu sygnalizacji bez wstrzymywania pracy programu głównego aplikacji. Przejdźmy, zatem do opisu najważniejszych, z punktu widzenia użytkownika, rejestrów, dzięki którym aplikacja użytkownika dostaje informację o bieżącym zdarzeniu z udziałem panelu dotykowego. Wykaz ważniejszych rejestrów zamieszczono w ramce.

Znaczenie bitów rejestrów konfiguracyjnych odpowiedzialnych za parametry pozostałych miejsc dotyku jest analogiczne jak rejestrów dla pierwszego miejsca dotyku, zaś ich adresy i funkcje przedstawiono w **tabeli 1**

Jak wspomniano, sterownik FT5X06 wyposażono w szereg dodatkowych rejestrów konfiguracyjnych, za pomocą których możemy wpływać na algorytm detekcji dotyku dla różnych warunków pracy obsługiwanego pojemnościowego panelu dotykowego oraz na progi aktywacji dostępnych trybów pracy, jednak w zdecydowanej większości przypadków typowych aplikacji, nie ma potrzeby ingerencji w domyślne ustawienia producenta układu. Inną sprawą jest, iż producent ten dość ogólnikowo opisuje znaczenie tychże rejestrów dla

Listing 2. Listing pliku nagłówkowego sterownika panelu dotykowego

```
//Structure that stores information for the touch
parameters
typedef struct
{
    uint16_t X, Y;  //Coordinates, resolution is:
896x640px
    uint8_t Flag;  //What has happened: Up, Down, Contact
    uint8_t Gesture;  //Gesture type: Move Up, Move Down,
Zoom In etc...
    uint8_t Ready;  //Set if touch is detected
} touchType;

//Structure that stores information for the displayed
element coordinates
typedef struct
{
    uint16_t X1;
    uint16_t Y1;
    uint16_t X2;
    uint16_t Y2;
} rectType;


#define CTP_WRITE_ADDR 0x70  //Touch panel write address
#define CTP_READ_ADDR 0x71  //Touch panel read address
#define CTP_INTERRUPT_NAME INT1_vect  //Symbolic ISR name
responsible for touch handling

#define CTP_INTERRUPT_PORT_REG PORTD
#define CTP_INTERRUPT_NR PD3
#define CTP_RESET_PORT_REG PORTC
#define CTP_RESET_DDR_REG DDRC
#define CTP_RESET_NR PC2
#define CTP_BUZZER_PORT_REG PORTD
#define CTP_BUZZER_DDR_REG DDRD
#define CTP_BUZZER_NR PD6


void CTPinit(void);
uint8_t isInRect(const rectType *Rectangle);
extern volatile touchType Touch;

#define DEVIDE_MODE_REG 0x00
#define GESTURE_ID_REG 0x01
#define GESTURE_MOVE_UP 0x10
#define GESTURE_MOVE_LEFT 0x14
#define GESTURE_MOVE_DOWN 0x18
#define GESTURE_MOVE_RIGHT 0x1C
#define GESTURE_ZOOM_IN 0x48
#define GESTURE_ZOOM_OUT 0x49
#define GESTURE_NO_GESTURE 0x00

#define TOUCH_POINTS_NR_REG 0x02
#define TOUCH1_XH_REG 0x03
#define TOUCH1_XL_REG 0x04
#define TOUCH1_YH_REG 0x05
#define TOUCH1_YL_REG 0x06
#define TOUCH_FLAG_PUT_DOWN 0x00
#define TOUCH_FLAG_PUT_UP 0x01
#define TOUCH_FLAG_CONTACT 0x02
#define TOUCH_FLAG_RESERVED 0x03

#define BUZZER_OFF CTP_BUZZER_PORT_REG |= (1<<CTP_BUZZER_
NR)
#define BUZZER_ON CTP_BUZZER_PORT_REG &= ~(1<<CTP_BUZZER_
NR)
#define START_BUZZER_TIMER TCCR0B |= (1<<CS02)|(1<<CS00)
//Tmer0, Prescaler = 1024
#define STOP_BUZZER_TIMER TCCR0B = 0
```



Rysunek 2. Przykładowe przebiegi na wyprowadzeniu /INT sterownika panelu dotykowego

pracy algorytmu detekcji jak i samego sterownika, co czyni je stosunkowo mało wartościowymi z punktu widzenia programisty. W tej chwili dysponujemy już wystarczającą wiedzą na temat samego sterownika panelu dotykowego by przejść do stosownego oprogramowania. Zanim jednak przejdę do szczegółów implementacyjnych, przedstawię podstawowe funkcje, dzięki którym możliwa jest komunikacja przy udziale interfejsu TWI mikrokontrolera, przy czym należy zaznaczyć, iż są to podstawowe implementacje poszczególnych funkcjonalności niewyposażone np. w mechanizm obsługi błędów, który moim zdaniem, nie jest elementem niezbędnym w tak prostych i niewielkich systemach mikroprocesorowych. Ciała funkcji odpowiedzialnych za obsługę interfejsu TWI pokazano na **listingu 1**.

Listing 3. Ciało funkcji odpowiedzialnej za inicjalizację sterownika panelu dotykowego

```
void CTPinit(void)
{
//I2C interface initialization (I2C frequency ~395kHz)
    TWI_Init();
//HOST INTERRUPT port pulled up
    CTP_INTERRUPT_PORT_REG |= (1<<CTP_INTERRUPT_NR);
//CTP Reset = 0 (CTP_RESET port as output)
    CTP_RESET_DDR_REG |= (1<<CTP_RESET_NR);
    _delay_ms(5);
//CTP Reset = 1
    CTP_RESET_PORT_REG |= (1<<CTP_RESET_NR);
//Configure Buzzer PORT and turn off the Buzzer (active 0)
    CTP_BUZZER_PORT_REG |= (1<<CTP_BUZZER_NR);
    CTP_BUZZER_DDR_REG |= (1<<CTP_BUZZER_NR);
//The falling edge of INTn generates an interrupt request
    EICRA |= (1<<ISC11);
    EIMSK |= (1<<INT1);  //External Interrupt INT1 is
enabled
//Enable Timer0 ISR responsible for buzzer handling. Each
touch generates short beep (appr.24ms)
    TIMSK0 |= (1<<TOIE0);  //Timer/Counter0 Overflow
Interrupt Enable
}
```

Listing 4. Ciało funkcji obsługi przerwania zewnętrznego odpowiedzialnej za odczyt zdarzeń panelu dotykowego

```
ISR(CTP_INTERRUPT_NAME)
{
    register uint8_t CTPreg;
    TWI_Start();
    TWI_WriteByte(CTP_WRITE_ADDR);
    TWI_WriteByte(GESTURE_ID_REG);
    TWI_Stop();
    _delay_us(5);
    TWI_Start();
    TWI_WriteByte(CTP_READ_ADDR);
    Touch.Gesture = TWI_ReadByte(ACK);  //Gesture type
    CTPreg = TWI_ReadByte(ACK);  //Touch points number -
not used
    CTPreg = TWI_ReadByte(ACK);  //Touch1 XH & Flag
    Touch.Flag = CTPreg >> 6;  //Touch1 Flag
    Touch.X = (CTPreg & 0x0F) << 8;
    Touch.X |= TWI_ReadByte(ACK);  //Touch1 XL
    Touch.Y =  (TWI_ReadByte(ACK) & 0x0F) << 8;  //Touch1
YH & Touch ID
    Touch.Y |= TWI_ReadByte(NACK);  //Touch1 YL
    TWI_Stop();
    if(Touch.Flag == TOUCH_FLAG_PUT_DOWN)
    {
        BUZZER_ON;
        START_BUZZER_TIMER;
    }
    Touch.Ready = 1;
}

ISR(TIMER0_OVF_vect)  //ISR responsible for buzzer
handling (buzzer off)
{
    BUZZER_OFF;
    STOP_BUZZER_TIMER;
}
```

Listing 5. Ciało funkcji narzędziowej do sprawdzania położenia współrzędnych panelu dotykowego

```
uint8_t isInRect(const rectType *Rectangle)
{
    register uint8_t Result = 0;
    if(Touch.X >= pgm_read_word(&Rectangle->X1) &&
Touch.X <= pgm_read_word (&Rectangle->X2))
        if(Touch.Y >= pgm_read_word (&Rectangle->Y1) &&
Touch.Y <= pgm_read_word (&Rectangle->Y2))     Result =
1;
    return Result;
}
```

Myślę, że nie wymagają one dodatkowego komentarza, gdyż ich nazwy są na tyle wymowne, iż nie pozostawiają wątpliwości, co do realizowanych przez nie funkcjonalności. Przejdźmy, zatem do funkcji bezpośrednio związanych z obsługą panelu dotykowego. Zanim jednak przedstawię listingi tychże funkcji warto poznać zawartość pliku nagłówkowego stosownego sterownika, który skonstruowano w taki sposób by bez lektury dokumentacji układu FT5X06 łatwo było się zorientować w realizowanej przezeń funkcjonalności. Dodatkowo zadeklarowano dwa, nowe typy strukturalne, które znacznie zwiększają czytelność programu obsługi jak również ułatwiają dostęp do zmiennych. Listing pliku nagłówkowego sterownika panelu dotykowego przedstawiono na **listingu 2**.

Na **listingu 3** przedstawiono ciało funkcji odpowiedzialnej za inicjalizację sterownika panelu dotykowego, zaś na **listingu 4** ciało funkcji obsługi przerwania zewnętrznego (w naszym przypadku INT1) odpowiedzialnej za odczyt zdarzeń panelu dotykowego (wyzwalane opadającym zboczem sygnału) oraz przerwania od przepełnienia Timera0, odpowiedzialnej za obsługę opcjonalnego buzzera.

Dodatkową, bardzo użyteczną funkcję narzędziową (zwłaszcza przy konstrukcji wszelkiego rodzaju graficznych interfejsów użytkownika), której zadaniem jest sprawdzanie czy odczytane koordynaty miejsca dotyku panelu dotykowego mieszczą się w zakresie obszaru określonego argumentami wywołania funkcji (prostokątny obszar określony parametrami współrzędnych X i Y lewego górnego i prawego, dolnego wierzchołka) przedstawiono na **listingu 5** (stosowne współrzędne należy zdefiniować jako zmienną strukturalną typu rectType w pamięci programu mikrokontrolera).

Funkcja ta zwraca wartość 1, jeśli bieżące koordynaty miejsca dotyku panelu dotykowego mieszczą się w zakresie obszaru określonego argumentami jej wywołania. Kończąc tematykę obsługi pojemnościowych paneli dotykowych przy udziale sterownika FT5X06 chciałbym szczególnie polecić Czytelnikom ten rodzaj układu i zintegrowany z nim, całkiem niedrogi wyświetlacz TFT jako doskonałe urządzenie peryferyjne dające nowe możliwości w konstruowaniu urządzeń mikroprocesorowych, dostępne dotychczas wyłącznie dla profesjonalistów. Co ciekawe, szukając informacji na temat tegoż układu, natknąłem się na źródła sterowników programowych dedykowanych właśnie jemu dla systemu operacyjnego Linux jak i innych, znanych systemów stosowanych w urządzeniach mobilnych, co dobitnie potwierdza jego atrakcyjność i nowoczesność zastosowanych w nim rozwiązań.

Robert Wołgajew, EP

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 76305, pass: 8741rnfv