ISIXRTOS v3 mini system operacyjny dla mikrokontrolerów rodziny M0/M3/M4/M7 (8)

Panel dotykowy

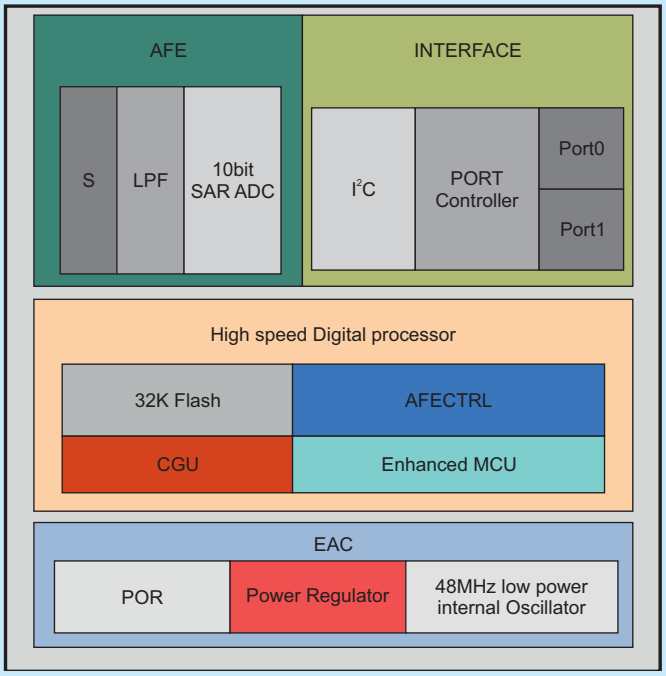
W poprzednim odcinku uruchomiliśmy wyświetlacz graficzny oraz przykładowe demo pokazujące możliwości graficzne biblioteki libgfx, jednak jak łatwo zauważyć brakuje nam jeszcze warstwy wejściowej, umożliwiającej wprowadzenie danych.

Wyświetlacz zestawu ewaluacyjnego STM32F469I-DISCO posiada pojemnościowy panel dotykowy, który możemy użyć jako źródło wprowadzania danych. Sercem panelu zintegrowanego z zestawem jest kontroler FT6x06 zapewniający następującą funkcjonalność:

- obsługa bezwzględnych współrzędnych x, y,
- obsługę gestów oraz wielodotyku,
- wsparcie dla auto kalibracji, której zadaniem jest dostosowanie panelu dotykowego do warunków środowiskowych.

Od strony elektrycznej dostęp do rejestrów kontrolera dotykowego możliwy jest za pośrednictwem magistrali I²C oraz dodatkowej linii przerwań zewnętrznych, umożliwiającej zgłaszanie zdarzeń bez konieczności ciągłego sprawdzania zawartości rejestrów układu. Budowę wewnętrzną układu przedstawiono na **rysunku 1**. Blok AFE stanowi część sprzętową i odpowiedzialny jest za interakcję z panelem dotykowym. Zawiera on blok generatora sygnałów szybkozmiennych oraz analogową część odbiorczą, umożliwiającą detekcję dotyku opartą o zmiany pojemności. Mikrokontroler z wbudowaną pamięcią ROM zawiera program zajmujący się obsługą panelu oraz odciążający procesor główny od monotonnej czynności ciągłego skanowania panelu. Komunikacja z hostem zewnętrznym odbywa się za pomocą magistrali I²C. Sposób podłączenia kontrolera do mikrokontrolera w zestawie ewaluacyjnym przedstawiono na **rysunku 2**.

Linie SDA i SCL panelu dotykowego zostały dołączone do portów PB8, PB9, które w trybie alternatywnym stanowią interfejs magistrali I2C1 mikrokontrolera. Linia przerwania informująca o wystąpieniu zdarzenia została podłączona do portu PJ5, który może być używany przez kontroler przerwań zewnętrznych EXTI. Sposób dołączenia panelu dotykowego do kontrolera odbywa się w sposób klasyczny, tak jak w przypadku każdego urządzenia z magistralą I²C.



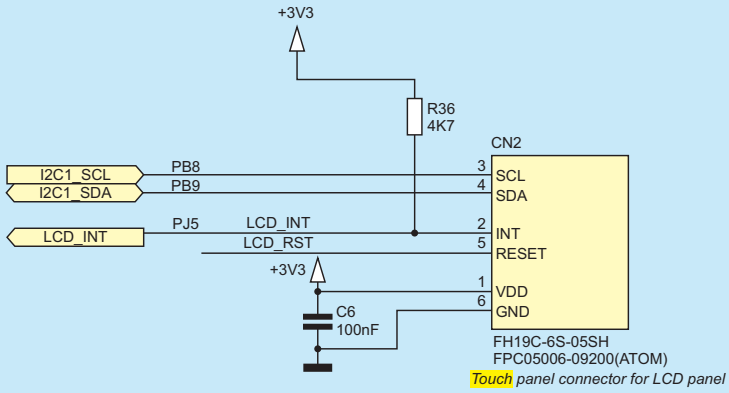
Rysunek 1. Budowa kontrolera dotykowego FT6x06

Oprogramowanie

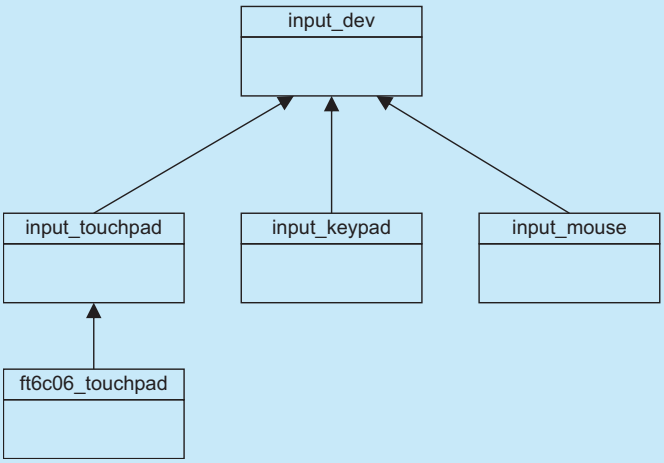
Za obsługę i generowanie zdarzeń od urządzeń peryferyjnych w bibliotece **libgfx** odpowiedzialny jest podsystem **input**. Hierarchię klas tego podsystemu przedstawiono na **rysunku 3**. Klasę bazową dla wszystkich klas pochodnych stanowi klasa **input_dev** umożliwiająca wstrzykiwanie zdarzeń do podsystemu graficznego. Z klasy bazowej dla urządzeń wejściowych dziedziczą poszczególne kategorie urządzeń takie jak: panel dotykowy, klawiatura oraz myszka. Tworząc sterownik urządzenia wejściowego dla biblioteki graficznej należy dziedziczyć z wybranej kategorii urządzenia wejściowego. W przypadku panelu dotykowego klasa **ft6x06_touchpad** odpowiedzialna za obsługę panelu dotykowego zestawu a jej jedynym publicznym interfejsem jest jej konstruktor oraz metoda *start()* umożliwiająca uruchomienie wątku odpowiedzialnego za obsługę kontrolera (**listing 1**).

Listing 1. Konstruktor oraz metoda umożliwiająca uruchomienie wątku odpowiedzialnego za obsługę kontrolera

```
public:
    //Constructor
    ft6x06_touch(const char disp_name[],
            periph::drivers::i2c_master& i2c,
            gui::frame& frame);
    // Start the main thread
    void start() noexcept;
```



Rysunek 2. Podłączenie panelu dotykowego do mikrokontrolera w zestawie STM32F469I-DISCO



Rysunek 3. Hierarchia klas podsystem.in.ut w bibliotece libgfx

Konstruktor przyjmuje szereg argumentów potrzebnych do działania sterownika i są to odpowiednio: nazwa panelu konfiguracyjnego, która jest później wykorzystywana przez podsystem DTS do odczytu parametrów konfiguracyjnych, referencja do obiektu kontrolera I²C oraz referencja do klasy frame, która odpowiedzialna jest za rysowanie elementów na ekranie. Za obsługę zdarzeń oraz odczyt danych z kontrolera odpowiedzialna jest metoda *thread()* (**listing 2**).

Metoda ta rozpoczyna pracę od wywołania metody *intialize()* odpowiedzialnej za konfiguracje układu ft6x06. W przypadku gdy inicjalizacja przebiegła pomyślnie wykonywana jest pętla nieskończona, która w cyklu 50 ms odczytuje z rejestrów konfiguracyjnych status kontrolera a jako wynik zwraca ilość zdarzeń jaka została wykryta. W przypadku wykrycia jakiegokolwiek zdarzenia wywoływana jest metoda *print_touch()* wypisująca na ekranie aktualnie wykryte zdarzeniem oraz metoda *report_touch()*. Zadaniem tej metody jest przekazanie zdarzeń dotykowych do klasy frame zarządzającej oknami na wyświetlaczu. Wspomniane wyżej funkcje jako jedyny argument przyjmują strukturę *touch_tag*, która opisuje zdarzenie dotykowe i pozwala później poszczególnym widżetom podjąć reakcję na poszczególne zdarzenia (**listing 3**). Do najważniejszych elementów tej struktury należą: pozycja bezwzględną wskazującą miejsce, gdzie ekran został dotknięty, szerokość obszaru dotknięcia oraz identyfikator gestu realizowanego za pomocą wielodotyku np. podniesienie okna, rozciągnięcie okna, zbliżenie oddalenie itp.

Wróćmy teraz na chwilę do funkcji inicjalizującej sterownik panelu dotykowego (**listing 4**). W metodzie tej w pierwszej kolejności wywoływana jest metoda *detect_device()*, której zadaniem jest stwierdzenie obecności dołączenia kontrolera do magistrali I²C. Działanie tej metody sprowadza się do odczytania zawartości rejestru identyfikacyjnego i porównaniu go z danym wzorcem. Jeśli wartość odczytana będzie różna od wartości wzorcowej lub nie uda się odczytać zawartości rejestru, zgłoszony będzie błąd, w wyniku czego wątek sterownika zakończy działanie. W przypadku, gdy

Listing 2. Metoda *thread()* odpowiedzialna za obsługę zdarzeń oraz odczyt danych z kontrolera

```
// Main thread for read input events
void ft6x06_touch::thread() {
    int ret;
    //Initialize touchpad hardware into the poll mode
    if((ret=initialize())) {
        dbg_err(„Unable to configure touchpad errno: %i”,ret);
        return;
    } else {
        dbg_info(„Touchpad initialized”);
    }
    for(touch_stat ts={};;ts={}) {
        isix::wait_ms(50);
        ret = get_state(ts);
        if(ret>0) {  //! If number of touches is greater than one
            //Print touch info
            print_touch(ts);
            //Report touch to the gui library
            report_touch(std::move(ts));
        } else if(ret<0) {
            dbg_err(„Touch screen failed”);
        }
    }
}
```

Listing 3. Strukturę touch_tag, która opisuje zdarzenie dotykowe

```
/** Touch screen tag  */
struct touch_tag {
    static constexpr auto max_touch = 2;
    unsigned char num_touches;              //! Number of touches
    unsigned short x[max_touch];            //! Position x
    unsigned short y[max_touch];            //! Position y
    unsigned char weight[max_touch];        //! Touch weight
    touchevents::touchevt eventid[max_touch];          //! Touch event id
    unsigned char area[max_touch];                     //! Touch area
    touchgestures::touchgests gestureid;               //! Gesture id
};
```

Listing 4. Funkcji inicjalizująca sterownik panelu dotykowego

```
//Initialize touchpad
int ft6x06_touch::initialize() noexcept
{
    int ret {};
    do {
        ret = detect_device();
        if(ret) break;
        const periph::dt::device_conf_base* base {};
        ret = periph::dt::get_periph_devconf(m_name,base);
        if(ret) break;
        const auto entry = reinterpret_
cast<const periph::display::fb_info*>(base);
        if(entry->n_layers<1) {
            ret = periph::error::inval;
            break;
        }
        ret = touch_enable(entry->layers[0].width, entry-
>layers[0].height);
        if(ret) break;
    } while(0);
    return ret;
}
```

Listing 5. Metodzie *touch_enable()*, której zadaniem jest odpowiednie skonfigurowanie układu

```
//Touch screen enable
int ft6x06_touch::touch_enable(
    unsigned short sizex,
    unsigned short sizey) noexcept
{
    if(sizex<sizey) {
        m_orientation = TS_SWAP_NONE;
    } else {
        m_orientation = TS_SWAP_XY | TS_SWAP_Y;
    }
    //Start the device
    int ret {};
    do {
        ret = calibrate();
        if(ret) break;
        ret = disable_it();
        if(ret) break;
    } while(0);
    return ret;
}
```

stwierdzimy iż kontroler został prawidłowo podłączony do magistrali, z bazy DTS odczytywana będzie aktualna rozdzielczość ekranu dotykowego. Rozdzielczość ta potrzebna jest metodzie *touch_enable()*, której zadaniem jest odpowiednie skonfigurowanie układu (**listing 5**).

W pierwszej kolejności sprawdzamy w jakiej orientacji pracuje wyświetlacz i w przypadku, gdy stwierdzimy, że znajduje się on w orientacji poziomej, ustawiane są dodatkowe bity konfiguracyjny zamieniające oś X z osią Y, które w późniejszym etapie będą przesłane do odpowiedniego rejestru. W kolejnym kroku wywoływana jest metoda *calibrate()*, której zadaniem jest kalibracja panelu dotykowego, tak aby dostosować go aktualnych warunków środowiskowych, a następnie wywoływana jest metoda *disable_int()*, której zadaniem jest wyłączenie zgłaszania przerwań na linii INT kontrolera, ponieważ w aktualnej implementacji sterownik pracuje w trybie odpytywania bez systemu przerwań.

Najważniejszą metodą wywoływaną w pętli głównej w cyklu 50 ms jest metoda *get_state()*, której zadaniem sprawdzenie czy wystąpiło zdarzenie oraz odczytanie informacji o typach zdarzeń dotykowych z rejestrów układu a następnie odpowiednie wypełnienie struktury przechowującej informację o zdarzeniu (**listing 6**). W pierwszej kolejności wywoływana jest metoda *get_touch()*, która z rejestrów kontrolera odczytuje ilość zdarzeń jaka została wykryta. W przypadku gdy liczba zdarzeń jest większa od 0,wówczas w pętli odczytywana jest pozycja (x,y) miejsca dotknięcia, jednak z uwagi iż wyświetlacz może być ustawiony w pozycji pionowej lub pionowej, odczytane z rejestrów układu współrzędne należy przeliczyć tak, aby odpowiadały one rzeczywistym współrzędnym ekranowym. Po przeliczeniu współrzędnych wywoływana jest metoda *get_info()*, której zadaniem jest odczytanie dodatkowych informacji o zdarzeniu takich jak m.in. typ gestu czy obszar w jakim nastąpiło zdarzenie. W kolejnym kroku w pętli switch..case następuje przeliczenie wartości rejestru zawierającego kod zdarzenia na identyfikatory zdarzeń występujące w bibliotece graficznej **libgfx**.

Mamy już gotowy sterownik umożliwiający zgłaszanie zdarzeń. Pozostało nam jeszcze utworzenie instancji klasy jako składowej prywatnej w klasie **tft_livedemo** przedstawionej w poprzednim odcinku, co zrealizowano bezpośrednio pliku *tft_livedemo.hpp*

```
gfx::drv::ft6x06_touch m_
touch { „display”, m_i2c, frame };
```

Jako pierwszy argument przekazujemy do konstruktora nazwę stanowiącą identyfikator urządzenia wyświetlacza, potrzebny do odczytania danych z konfiguracji DTS, referencję do sterownika magistrali I²C oraz referencję do klasy frame biblioteki graficznej, która odpowiedzialna jest za zarządzanie oknem głównym biblioteki graficznej.

Aby uruchomić powyższy przykład należy podłączyć do komputera zestaw stm32f469i-disco z wykorzystaniem kabla mini USB oraz pobrać kod źródłowy z repozytorium wpisując polecenie:

```
git clone -b pub/ep1119 –recursive https://www.boff.
pl/cgit/public/isixsamples
```

W kolejnym kroku należy skompilować program oraz zaprogramować płytkę ewaluacyjną tak powstałym kodem maszynowym, co możemy zrealizować wydając następujące polecenia:

```
waf configure --crystal-hz=8000000 --debug waf
waf
waf program
```

Listing 6. Metoda *get_state()*, której zadaniem jest sprawdzenie czy wystąpiło zdarzenie oraz odczytanie informacji o typach zdarzeniach

```
// Get gesture state
int ft6x06_touch::get_state(touch_stat& stat) noexcept {
    using namespace detail::regs::ft6x06;
    using namespace gfx::input;
    int ts_status;
    do {
        /* Check and update the number of touches active detected */
        ts_status = detect_touch();
        if (ts_status<0) break;
        stat.num_touches = ts_status;
        ts_status = 0;
        if (stat.num_touches>0) {
            uint16_t Raw_x[c_max_nb_touch];
            uint16_t Raw_y[c_max_nb_touch];
            for (int index = 0; index < stat.num_touches; index++){
                /* Get each touch coordinates */
                ts_status = get_xy( (Raw_x[index]), (Raw_y[index]));
                if(ts_status) break;
                if (m_orientation & TS_SWAP_XY) {
                    auto tmp = Raw_x[index];
                    Raw_x[index] = Raw_y[index];
                    Raw_y[index] = tmp;
                }

                if (m_orientation & TS_SWAP_X) {
                    Raw_x[index] = FT_6206_MAX_WIDTH - 1 - Raw_x[index];
                }

                if (m_orientation & TS_SWAP_Y) {
                    Raw_y[index] = FT_6206_MAX_HEIGHT - 1 - Raw_y[index];
                }

                auto xDiff = Raw_x[index] > m_x[index] ?
                    (Raw_x[index] - m_x[index]) : (m_x[index] - Raw_x[index]);

                auto yDiff = Raw_y[index] > m_y[index] ?
                    (Raw_y[index] - m_y[index]) : (m_y[index] - Raw_y[index]);

                if ((xDiff + yDiff) > 5) {
                    m_x[index] = Raw_x[index];
                    m_y[index] = Raw_y[index];
                }

                stat.x[index] = m_x[index];
                stat.y[index] = m_y[index];
                uint32_t weight {};
                uint32_t area {};
                uint32_t event {};
                ts_status = get_info(index, weight, area, event);
                if(ts_status) break;
                stat.weight[index] = weight;
                stat.area[index] = area;

                /* Remap touch event */
                switch (event) {
                case FT6206_TOUCH_EVT_FLAG_PRESS_DOWN:
                    stat.eventid[index] = touchevents::press_down;
                    break;
                case FT6206_TOUCH_EVT_FLAG_LIFT_UP:
                    stat.eventid[index] = touchevents::lift_up;
                    break;
                case FT6206_TOUCH_EVT_FLAG_CONTACT:
                    stat.eventid[index] = touchevents::contact;
                    break;
                case FT6206_TOUCH_EVT_FLAG_NO_EVENT:
                    stat.eventid[index] = touchevents::undefined;
                    break;
                default:
                    ts_status = error::error_not_supported;
                    break;
                }
            }

            /* Get gesture Id */
            ts_status = get_gesture_code();
            if(ts_status<0) break;
        }
    } while (0) ;
    return ts_status?ts_status:stat.num_touches;
}
```

Równocześnie do wirtualnego portu szeregowego utworzonego przez programator należy podłączyć dowolny program terminalowy oraz ustawiając następujące parametry transmisji: 115200,n,8,1.

Po wykonaniu powyższych czynności na wyświetlaczu powinno pojawić się okno z demonstracyjnymi widżetami biblioteki graficznej. Dotykając teraz poszczególnych elementów na ekranie na konsoli szeregowej będziemy mogli zaobserwować informację o pozycji dotknięcia, obszarze dotknięcia oraz ewentualnych gestach. Jednak nadal interfejs dotykowy w żaden sposób nie będzie reagował na zdarzenia. O tym jak zrobić, aby wspomniane elementy reagowały na dotyk, dowiemy się z kolejnego odcinka, gdy zostanie przedstawione więcej szczegółów na temat biblioteki **libgfx**.

Lucjan Bryndza, EP
lucjan.bryndza@boff.pl

Poprzednie części kursu i dodatkowe materiały dostępne są na stronie **www.media.avt.pl**