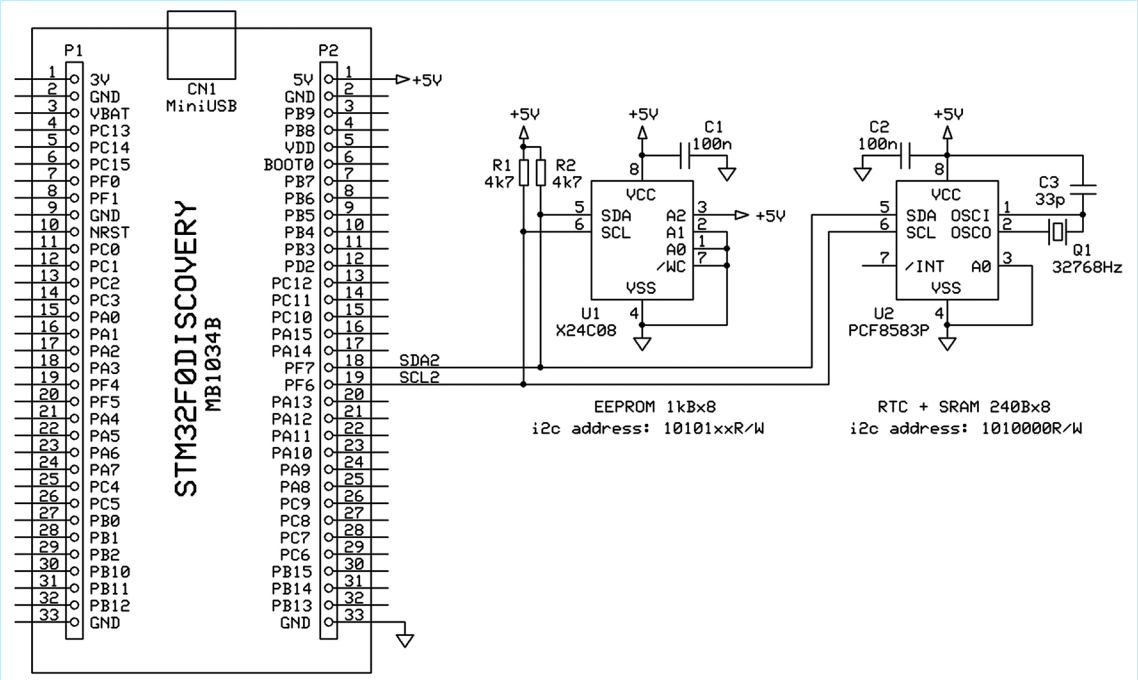
Programator/debugger ST-Link

Programowanie pamięci zewnętrznych w systemie z mikrokontrolerem STM32

Podczas konstruowania systemów mikroprocesorowych wyposażonych w zewnętrzne układy pamięci często pojawia się potrzeba podglądu i modyfikacji ich zawartości. O ile z dostępem do wewnętrznej pamięci mikrokontrolera najczęściej nie ma problemu – realizuje to programator / debugger, o tyle z dostępem do zewnętrznych pamięci dołączonych do mikrokontrolera nie jest już tak łatwo. Można oczywiście posiłkować się specjalnie tworzonymi do tego celu funkcjami, umieszczanymi w pisanym programie, które wspomagają debugowanie zawartości zewnętrznej pamięci, np. przez łącze szeregowe RS, jednak nie zawsze jest to możliwe i wygodne, zwłaszcza gdy zastosowany mikrokontroler nie ma wolnej pamięci programu na dodatkowe funkcje. W przypadku mikrokontrolerów STM32 istnieje jednak alternatywa. Jest nią program narzędziowy o nazwie ST-Link Utility, stworzony przez firmę STMicroelectronics do obsługi programatora/debuggera ST-Link.

Dla produkowanych przez siebie mikrokontrolerów rodziny STM32 firma STMicroelectronics dostarcza programator/debugger *ST-Link* [1] oraz obsługujący go program narzędziowy o nazwie *ST-Link Utility* [2]. Podstawowym przeznaczeniem tego programu jest programowanie z poziomu komputera PC wewnętrznej pamięci Flash mikrokontrolerów STM32 oraz ustawianie ich bitów konfiguracyjnych zgrupowanych w tzw. *Option Bytes*. Na tym jednak nie kończą się możliwości programu *ST-Link Utility*. Oferuje on bowiem dodatkową, bardzo przydatną w praktyce konstruktora funkcję, jaką jest możliwość zapisu i odczytu w systemie zewnętrznych pamięci dołączonych do mikrokontrolera STM32. Funkcja ta kryje się w menu programu pod nazwą *External Loader*.

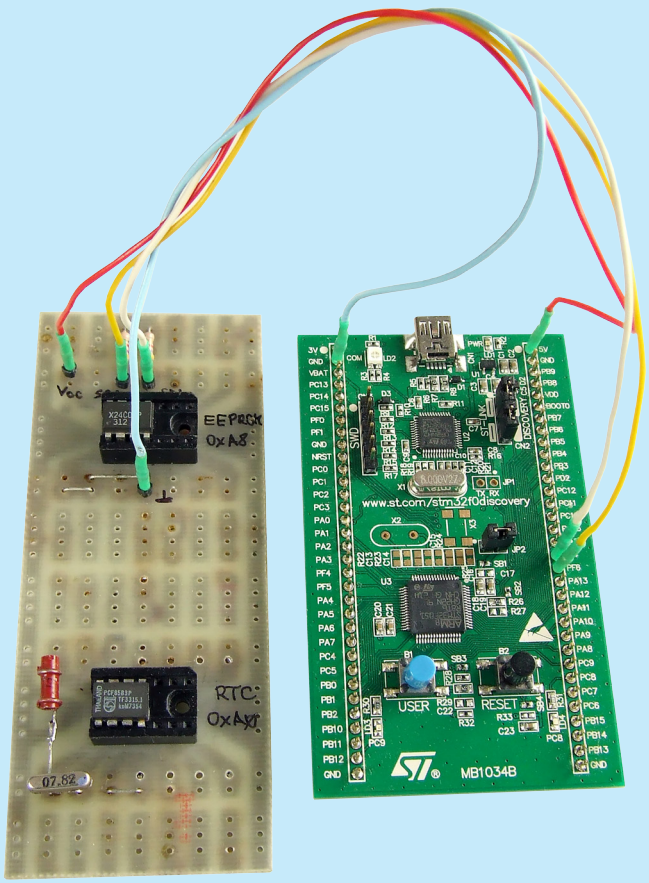
Obsługa zewnętrznej pamięci jest realizowana przez program *ST-Link Utility* w niezwykle interesujący sposób. Otóż program ten ładuje do pamięci SRAM mikrokontrolera kod maszynowy funkcji realizującej zadaną operację, np. odczytu danych z zewnętrznej pamięci. Następnie kod ten jest wykonywany przez mikrokontroler, a uzyskane w trakcie tego procesu dane są pobierane z mikrokontrolera przez program *ST-Link Utility* i prezentowane użytkownikowi na ekranie komputera PC. Mechanizm ten jest na tyle uniwersalny, że może być użyty nie tylko do odczytu i zapisu pamięci, lecz także do obsługi innych dołączonych do mikrokontrolera



Rysunek 1. Schemat układu demonstracyjnego obsługi pamięci zewnętrznych przez program ST-Link Utility

Tabela 1. Funkcje obsługi pamięci EEPROM 24C01/02/04/08/16				
Lp	Nazwa Funkcji	Parametry wywołania	Zwracana wartość	Opis
1	i2c_eeprom_Init	brak	brak	Inicjalizacja układu peryferyjnego I2Cx mikrokontrolera STM32F051 oraz używanych przez ten układ linii GPIO. Układ peryferyjny I2Cx jest konfigurowany do transmisji danych magistralą I²C ze standardową prędkością 100 kbit/s
2	i2c_eeprom_CheckReady	brak	1 – układ gotowy 0 – układ zajęty lub błąd	Kontrola gotowości pamięci EEPROM do wykonania operacji. Kontrola gotowości jest realizowana w drodze cyklicznych wywołań pamięci EEPROM i analizy bitu ACK/NAK jej odpowiedzi
3	i2c_eeprom_ReadBlock	pBuffer – wskaźnik do bufora na dane readAddr – adres początkowy odczytywanych danych dataCount – liczba odczytywanych danych	1 – odczyt poprawny 0 – błąd	Odczyt z pamięci EEPROM bloku danych do bufora w pamięci SRAM
4	i2c_eeprom_WriteBlock	pBuffer – wskaźnik do bufora z danymi writeAddr – adres początkowy zapisu danych w pamięci dataCount – liczba zapisywanych danych	1 – zapis poprawny 0 – błąd	Zapis w pamięci EEPROM bloku danych z bufora w pamięci SRAM. Zapisywane dane są automatycznie stronicowane, zgodnie ze zdefiniowanym rozmiarem strony pamięci EEPROM.

układów, do których dostęp odbywa się przez zapis i odczyt rejestrów, czyli np. do zegara RTC. Jak nietrudno się domyślić, ładowany do pamięci SRAM mikrokontrolera kod, nazywany dalej sterownikiem pamięci zewnętrznej, jest w zasadzie specjalizowanym programem dla mikrokontrolera STM32, zależnym od typu mikrokontrolera, rodzaju i typu zewnętrznej pamięci oraz sposobu jej podłączenia do mikrokontrolera. W folderze *ExternalLoader* programu *ST-Link Utility* znajduje się kod sterowników realizujących obsługę pamięci zewnętrznych w produkowanych przez firmę STMicroelectronics zestawach rozwojowych mikrokontrolerów STM32. W przypadku, gdy konstruowany układ wykorzystuje moduł pamięci, który nie był stosowany w jednym z zestawów rozwojowych STM32 lub też pamięć ta jest inaczej podłączona niż w zestawie ewaluacyjnym, sterownik obsługi tej pamięci dla programu *ST-Link Utility* musi zostać napisany przez użytkownika. Zagadnienie tworzenia sterowników jest omówione w rozdziale 3.9 instrukcji obsługi programu *ST-Link Utility*. Niestety, opis ten jest dość skromny i ogranicza się zaledwie do kilkunastu zdań. Z tego też względu podjęto próbę zaprezentowania na przykładowym układzie z mikrokontrolerem STM32, jak wygląda pisanie takiego sterownika.



Fotografia 2. Układ wykorzystywany podczas prac nad sterownikiem pamięci zewnętrznych dla programu ST-Link Utility

Układ demonstracyjny

Schemat ideowy układu, dla którego zostanie przedstawiony proces tworzenia sterownika obsługi pamięci zewnętrznych dla programu *ST-Link Utility*, przedstawia **rysunek 1**. Układ ten składa się z modułu STM32F0DISCOVERY [3], do którego dołączono pamięć EEPROM typu X24C08 oraz zegar RTC typu PCF8583. Moduł STM32F0DISCOVERY jest tanim zestawem ewaluacyjnym dla mikrokontrolera STM32F051, wielokrotnie opisywanym na łamach „Elektroniki Praktycznej". Warto przypomnieć, że moduł ten jest wyposażony w mikrokontroler typu STM32F051R8T6 z rdzeniem Cortex-M0, mający 64 kB pamięci Flash i 8 kB pamięci SRAM. W module Discovery wbudowano również układ programatora/debuggera *ST-Link V2*.

Układ X24C08 [4] jest pamięcią EEPROM o pojemności 8 kbit i organizacji 1024×8 bitów, wyposażoną w interfejs I²C. Układ PCF8583 [5] jest kombinacją zegara RTC i pamięci SRAM, zorganizowanych jako pamięć o pojemności 256 bajtów. Pierwsze 16 komórek tej pamięci to rejestry zegara RTC, natomiast kolejne 240 bajty to komórki pamięci SRAM. Podobnie jak pamięć EEPROM, układ PCF8583 jest wyposażony w interfejs I²C. W prezentowanym przykładzie oba układy są dołączone do wspólnej magistrali obsługiwanej przez sprzętowy układ interfejsu I2C2 mikrokontrolera STM32F051 w module Discovery. Linie SCL i SDA

tego interfejsu są wyprowadzone na porty PF6 i PF7 mikrokontrolera i dostępne na wyprowadzeniach 19 i 18 złącza P2 modułu Discovery. Pamięć EEPROM jest skonfigurowana do pracy na magistrali I²C pod adresem bazowym 0xA8, natomiast układ PCF8583 pracuje pod adresem bazowym 0xA0.

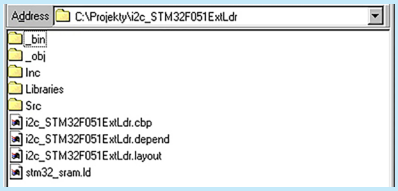
Ponieważ zastosowana w prezentowanym przykładzie pamięć X24C08 jest przystosowana do zasilania napięciem +5 V, zdecydowano się na zasilanie wszystkich układów dołączonych do magistrali I²C napięciem +5 V pobieranym z modułu Discovery. Nie stanowi to problemu, ponieważ linie PF6 i PF7 mikrokontrolera STM32F051 tolerują napięcie+5 V, a układ zegara PCF8583 może pracować z zasilaniem z zakresu 2,5…6,0 V. Oczywiście, w wypadku użycia niskonapięciowej wersji pamięci EEPROM, mogącej pracować przy napięciu +3,3 V, cały układ demonstracyjny może być zasilony bezpośrednio z tego napięcia. Jest ono dostępne na wyprowadzeniu nr 1 złącza P1 modułu Discovery. Zmontowany układ przedstawiono na **fotografii 2**.

Kod sterownika pamięci zewnętrznych

Ponieważ w układzie demonstracyjnym znajdują się dwa różne typy pamięci zewnętrznych, do ich obsługi przez program *ST-Link Utility* konieczne jest napisanie dwóch oddzielnych sterowników. Na szczęście algorytmy odczytu i zapisu danych z/do pamięci X24C08 oraz zegara PCF8583 są na tyle zbliżone, że możliwe jest wygenerowanie kodu maszynowego sterownika dla każdego z powyższych układów na podstawie tego samego kodu źródłowego, w którym bezpośrednio przed kompilacją modyfikowane są jedynie parametry konfiguracyjne układu pamięci, takie jak rozmiar pamięci, jej adres bazowy, itp.

Pierwszym krokiem w pisaniu sterownika zewnętrznej pamięci jest wykonanie za pomocą środowiska programistycznego pustego projektu dla mikrokontrolera STM32F051. Projekt ten powinien być linkowany do pamięci SRAM, poczynając od adresu 0x20000004. W przeciwieństwie do typowych programów pisanych dla mikrokontrolerów w języku C, tworzony projekt nie wymaga dołączania plików z kodem startowym. Przy tworzeniu pustego projektu sterownika najprościej wzorować się na jednym z przykładowych projektów sterowników pamięci zewnętrznych SRAM i Flash, które są dostarczane razem z programem *ST-Link Utility* i znajdują się w folderze *ExternalLoader*. Projekty te są przystosowane do kompilacji w kilku popularnych środowiskach IDE, więc nie powinno być problemów z importem przykładowego projektu do używanego środowiska programistycznego. Dołączony do artykułu kod sterowników pamięci X24C08 i zegara PCF8583 był kompilowany przy użyciu pakietu GCC. Strukturę tego projektu przedstawia **rysunek 3**. Foldery *Inc* i *Src* zawierają odpowiednio pliki nagłówkowe i kod źródłowy uniwersalnego sterownika układów X24C08 i PCF8583. W folderze *Libraries* znajdują się z kolei pliki bibliotek CMSIS dla rdzeni Cortex-M oraz dla mikrokontrolerów rodziny STM32F0xx. W folderach *_obj* i *_bin* umieszczany jest natomiast kod wynikowy.

Do wykonania sterowników umożliwiających programowi *ST-Link Utility* obsługę układów X24C08 i PCF8583 potrzebne są funkcje realizujące inicjalizację mikrokontrolera i jego układów peryferyjnych obsługujących interfejs I²C, jak również funkcje odczytu i zapisu danych z/do układów dołączonych do magistrali I²C. Można do tego celu użyć gotowych funkcji z biblioteki *STM32F0xx_StdPariph_Lib* lub funkcji z nowszej biblioteki *HAL – Hardware Abstraction Layer*. Niestety, obie biblioteki mają opinię „pamięciożernych”, co może mieć znaczenie dla mikrokontrolerów wyposażonych w pamięć SRAM o mniejszej pojemności. Należy pamiętać, że w tej pamięci musi się zmieścić nie tylko oprogramowanie sterownika, ale także bufory danych tworzone przez program *ST-Link Utility* do komunikacji ze sterownikiem. Co prawda zainstalowana w module Discovery wersja mikrokontrolera STM32F051 ma 8 kB pamięci SRAM i raczej nie należy spodziewać się problemów, jednak przy tworzeniu prezentowanego sterownika założono, że będzie on działał również na wersji



Rysunek 3. Struktura projektu sterowników pamięci zewnętrznych dla programu ST-Link Utility

mikrokontrolera STM32F051 z pamięcią SRAM o pojemności 4 kB. Z tego też powodu zdecydowano się na napisanie własnej funkcji inicjalizacji układu peryferyjnego obsługującego magistralę I²C, a także własnych funkcji odczytu i zapisu bloku danych z/do układu dołączonego do magistrali I²C. Stworzone na potrzeby projektu funkcje operują bezpośrednio na rejestrach mikrokontrolera i wykorzystują automatyczny tryb transmisji danych układu komunikacyjnego I2Cx mikrokontrolera STM32F051, przez co odznaczają się małym zapotrzebowaniem na pamięć programu i pamięć danych. W omawianym projekcie sterownika funkcje te zostały zgrupowane w pliku źródłowym *i2c_eeprom.c*, który znajduje się w folderze *Src* prezentowanego projektu. Szczegółowe omawianie kodu źródłowego tych funkcji wykracza poza ramy artykułu. Osoby zainteresowane tym tematem powinny przeanalizować kod zawarty w pliku *i2c_eeprom.c*, który znajduje się w materiałach dołączonych do artykułu. Listę funkcji zdefiniowanych w pliku *i2c_eeprom.c* zestawiono w **tabeli 1**.

Umieszczone w pliku *i2c_eeprom.c* funkcje obsługi pamięci EEPROM są uniwersalne i zapewniają obsługę przez mikrokontroler STM32F051 układu pamięci EEPROM typu 24Cxx mieszczącej od 128 B do 2 kB dołączonej do dowolnego układu peryferyjnego I2Cx mikrokontrolera. Konfiguracja używanego układu I2Cx mikrokontrolera, jego linii wyjściowych, jak również parametrów pamięci EEPROM, tj. jej pojemności, wielkości strony pamięci oraz adresu na magistrali I²C jest realizowana przez odpowiednie deklaracje *#define* zebrane w pliku nagłówkowym *i2c_eeprom.h*, znajdującym się w folderze *Inc* projektu.

Układ zegara RTC typu PCF8583 można traktować jako pamięć szeregową I²C o pojemności 256 B. Ponieważ wymiana danych z tym układem przebiega praktycznie identycznie, jak w wypadku pamięci EEPROM, nie ma sensu tworzenie przeznaczonych dla tego układu funkcji obsługi, gdyż układ PCF8583 może być obsługiwany przez funkcje napisane dla pamięci EEPROM. W celu zminimalizowania przez funkcję zapisu *i2c_eeprom_WriteBlock()* zbędnej kontroli gotowości układu PCF8583 do wykonania kolejnej operacji, należy jednak przed skompilowaniem projektu dla układu PCF8583 ustawić rozmiar zapisywanej strony pamięci (parametr *EEPROM_PAGE_LEN*) na 128 B. Optymalnym rozwiązaniem byłoby zadeklarowanie dla tego układu rozmiaru strony pamięci równego 256 B, jednak konstrukcja funkcji *i2c_eeprom_WriteBlock()* nie dopuszcza takiej możliwości. Poza zmianą rozmiaru strony pamięci nie należy oczywiście zapomnieć, w przypadku kompilacji projektu dla układu PCF8583, o ustawieniu rozmiaru pamięci (parametr *EEPROM_SIZE*) na 256 B i zmianie adresu układu na magistrali I²C (parametr *EEPROM_HWD_ADDR*) na 0xA0.

Dysponując funkcjami inicjalizacji układu peryferyjnego I2Cx mikrokontrolera oraz zapisu i odczytu danych do/z układów dołączonych do magistrali i2c, można przystąpić do pisania właściwego kodu sterownika pamięci zewnętrznych dla programu *ST-Link Utility*. Sterownik ten składa się z trzech plików: pliku nagłówkowego *Dev_Inf.h* oraz dwóch plików z kodem sterownika: *Dev_inf.c* i *Loader_Src.c*.

Plik nagłówkowy *Dev_Inf.h* jest umieszczony w folderze *Inc* prezentowanego projektu i zawiera definicje możliwych rodzajów pamięci oraz definicje struktur opisujących cechy pamięci, którą obsługuje sterownik. Zawartość pliku nagłówkowego przedstawia **listing 1**. Jak widać, autorzy programu *ST-Link Utility* przewidzieli możliwość obsługi 10 różnych rodzajów pamięci zewnętrznych. Rodzaj zadeklarowanej pamięci ma

Listing 1: Plik nagłówkowy *Dev_inf.h* sterownika pamięci zewnętrznych

```
#define  MCU_FLASH   1
#define  NAND_FLASH  2
#define  NOR_FLASH   3
#define  SRAM        4
#define  PSRAM       5
#define  PC_CARD     6
#define  SPI_FLASH   7
#define  I2C_FLASH   8
#define  SDRAM       9
#define  I2C_EEPROM  10
#define SECTOR_NUM 10    // Max Number of Sector types

struct DeviceSectors
{
    unsigned longSectorNum;  // Number of Sectors
    unsigned longSectorSize;  // Sector Size in Bytes
};

struct StorageInfo
{
    char DeviceName[100];  // Device Name and Description
    unsigned short DeviceType;  // Device Type: ONCHIP, EXT8BIT, EXT16BIT
    unsigned longDeviceStartAddress;  // Default Device Start Address
    unsigned longDeviceSize;  // Total Size of Device
    unsigned longPageSize;  // Programming Page Size
    unsigned charEraseValue;  // Content of Erased Memory
    struct DeviceSectors  sectors[SECTOR_NUM];
};
```

jednak znaczenie przede wszystkim informacyjne i tylko w niewielkim stopniu wpływa na algorytm obsługi pamięci przez program *ST-Link Utility*. Główna różnica polega na kasowaniu, bądź nie, pamięci zewnętrznej przed zapisem nowych danych. W przypadku pisania sterownika dla układu, którego nie ma na liście, wystarczy więc wybranie z tej listy rodzaju pamięci jak najbardziej zbliżonej sposobem działania.

Plik *Dev_Inf.c* (listing 1) jest umieszczony w folderze *Src* prezentowanego projektu i zawiera strukturę *StorageInfo* z opisem pamięci obsługiwanej przez sterownik. Ta struktura jest linkowana do wydzielonej sekcji programu i informacje w niej zawarte są wykorzystywane przez program *ST-Link Utility* do wyboru właściwego sterownika oraz algorytmu kasowania i zapisu pamięci zewnętrznej. Struktura *StorageInfo* obejmuje 7 pól.

Pierwsze pole *DeviceName* zawiera tekstowy opis sterownika. Opis ten jest wyświetlany przez program *ST-Link Utility* podczas wyboru sterownika. Warto więc zadbać, aby opis ten był pełny i jednoznacznie identyfikował nie tylko pamięć, ale i jej otoczenie. Dzięki temu uniknie się sytuacji, w której w programie *ST-Link Utility* są zainstalowane np. dwa sterowniki pamięci 24C08, różniące się tylko numerem wykorzystywanego w mikrokontrolerze STM32F051 układu I2Cx i opisane jako „24C08". Przy tak niejednoznacznym opisie nie będzie możliwe wybranie właściwego sterownika z listy dostępnych sterowników pamięci zewnętrznych wyświetlanej w programie *ST-Link Utility*.

Drugie pole *DeviceType* struktury *StorageInfo* zawiera definicję rodzaju zewnętrznej pamięci. Jak już wcześniej wspomniano, zadeklarowany rodzaj pamięci determinuje sposób obsługi przez program *ST-Link Utility* pamięci zewnętrznej przy zapisie danych. W przypadku układu X24C08 pole to przyjmuje wartość I2C_EEPROM z predefiniowanej listy rodzajów pamięci w pliku nagłówkowym *Dev_Inf.h*. W przypadku zegara RTC typu PCF8583, ponieważ tego rodzaju układu nie ma na wyżej wspominanej liście, polu temu przypisano najbardziej zbliżony rodzaj pamięci z listy, tj. *SRAM*.

Kolejne dwa pola struktury *DeviceStartAddress* i *DeviceSize* definiują odpowiednio adres bazowy pamięci zewnętrznej w przestrzeni adresowej mikrokontrolera STM32 oraz rozmiar tej pamięci. Każde odwołanie w programie *ST-Link Utility* do adresu w zakresie od *DeviceStartAddress* do *DeviceStartAddress+DeviceSize-1* będzie skutkowało tym, że dostęp do pamięci będzie realizowany za pośrednictwem danego sterownika. W prezentowanym przykładzie przyjęto, że pamięć EEPROM będzie widoczna w przestrzeni adresowej mikrokontrolera STM32 pod adresem bazowym 0x60000000, natomiast zegar RTC – pod adresem bazowym 0x60001000. Adres 0x60000000 jest początkiem obszaru w przestrzeni adresowej mikrokontrolerów ARM, w którym typowo są umieszczane pamięci zewnętrzne. Pole *DeviceSize* przyjmuje w przypadku układów X24C08 i PCF8583 wartości odpowiednio 0×400 (tj. 1 kB) i 0×100 (tj. 256 B).

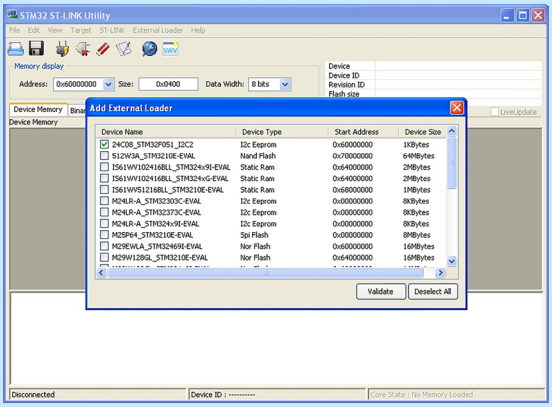
Pole *PageSize* określa rozmiar strony pamięci zapisywanej w pojedynczym cyklu. W przypadku pamięci EEPROM typu X24C08 firmy Xicor zapisywana strona pamięci ma długość 16 B. Należy jednak pamiętać, że układy pamięci 24C08 innych producentów mogą różnić się rozmiarem zapisywanej strony pamięci. Dlatego też każdorazowo należy zweryfikować tę wartość w karcie katalogowej używanego układu. Zegar RTC nie

Listing 2. Struktura *StorageInfo* z opisem pamięci EEPROM X24C08

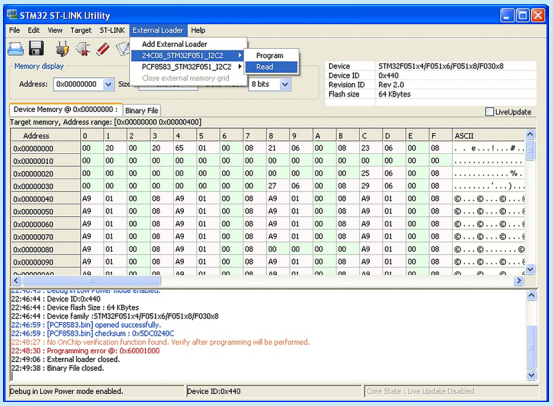
```
struct StorageInfo const StorageInfo __attribute__((section(„.DevInfoData"))) = {
    „PCF8583_STM32F051",  // Device Name + version number
    SRAM,  // Device Type
    0x60001000,  // Device Start Address
    0x00000100,  // Device Size in Bytes: 256 Bytes
    0x00000080,  // Programming Page Size: 128 Bytes
    0xFF,  // Initial Content of Erased Memory
// Specify Size and Address of Sectors
    {
        {0x00000001, 0x00000100},
        {0x00000000, 0x00000000}
    }
};
```

Listing 3: Struktura *StorageInfo* z opisem zegara RTC PCF8583

```
struct StorageInfo const StorageInfo __attribute__((section(„.DevInfoData"))) =  {
    „24C08_STM32F051",  // Device Name + version number
    I2C_EEPROM,  // Device Type
    0x60000000,  // Device Start Address
    0x00000400,  // Device Size in Bytes: 1 kBytes
    0x00000010,  // Programming Page Size: 16 Bytes
    0xFF,  // Initial Content of Erased Memory
// Specify Size and Address of Sectors
    {
        {0x00000001, 0x00000400},
        {0x00000000, 0x00000000}
    }
};
```



Rysunek 4. Wybór aktywnych sterowników pamięci zewnętrznych w programie ST-Link Utility



Rysunek 5. Menu obsługi pamięci zewnętrznej 24C08 w programie ST-Link Utility

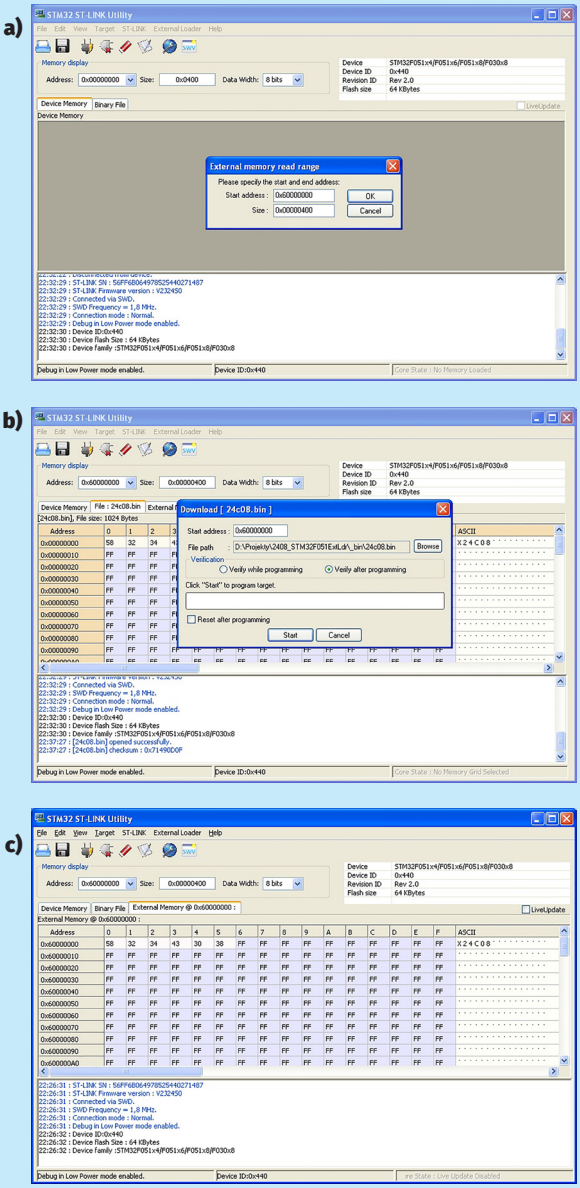
ma takich ograniczeń i cały ten układ może być zapisany w jednym cyklu. Jak już jednak wcześniej wspomniano, ze względu na ograniczenia funkcji *i2c_eeprom_WriteBlock()* zapisu bloku danych przyjęto, że rozmiar strony pamięci zegara RTC wynosi 128 B. Układ ten będzie więc zapisywany w dwóch cyklach.

Ostatnie dwa pola struktury *StorageInfo* zawierają zawartość komórki pamięci po skasowaniu (pole *EraseValue*) oraz listę sektorów pamięci i ich rozmiarów (pole *sectors*). Ponieważ w prezentowanym przykładzie są używane układy, które nie wymagają wcześniejszego kasowania przed zapisem nowych danych, pola te nie mają praktycznego znaczenia. Zarówno w przypadku układu X24C08, jak i PCF8583 przyjęto, że rozmiar sektora jest równy pojemności układu. Wartości (0x00000000, 0x00000000) są znacznikami końca listy sektorów.

Zawartość wypełnionej struktury *StorageInfo* dla pamięci 24C08 oraz zegara PCF8583 przedstawiają odpowiednio **listingi 2** i **3**.

Tabela 2: Funkcje sterownika pamięci zewnętrznych dla programu ST-Link Utility

Lp.	Nazwa Funkcji	Parametry wywołania	Zwracana wartość	Opis
1	Init	brak	1 – inicjalizacja mikrokontrolera i układów peryferyjnych zakończona sukcesem 0 – błąd	Inicjalizacja systemu generowania sygnałów zegarowych mikrokontrolera oraz inicjalizacja układów peryferyjnych i linii GPIO obsługujących pamięć zewnętrzną. Obsługiwane rodzaje pamięci: Funkcja obowiązkowa dla wszystkich rodzajów pamięci.
2	Read	Address – adres początkowy odczytywanych danych, Size – liczba odczytywanych danych (w bajtach) Buffer – wskaźnik do bufora na dane	1 – odczyt danych zakończony sukcesem 0 – błąd	Odczyt danych z pamięci zewnętrznej. Obsługiwane rodzaje pamięci: Funkcja obowiązkowa dla wszystkich rodzajów pamięci poza pamięciami SRAM, PSRAM i NOR_FLASH.
3	Write	Address – adres początkowy zapisu danych, Size – liczba zapisywanych danych (w bajtach) Buffer – wskaźnik do bufora z danymi	1 – zapis danych zakończony sukcesem 0 – błąd	Zapis danych w pamięci zewnętrznej. Obsługiwane rodzaje pamięci: Funkcja obowiązkowa dla wszystkich rodzajów pamięci poza pamięciami SRAM i PSRAM.
4	MassErase	brak	1 – kasowanie pamięci zakończone sukcesem 0 – błąd	Całkowite kasowanie pamięci zewnętrznej. Obsługiwane rodzaje pamięci: Funkcja obowiązkowa dla wszystkich rodzajów pamięci poza pamięciami SRAM, PSRAM i NOR_FLASH.
5	SectorErase	EraseStartAddress – adres początkowy kasowanego obszaru EraseEndAddress – adres końcowy kasowanego obszaru	1 – operacja zakończona sukcesem 0 – błąd	Kasowanie sektora w pamięci zewnętrznej. Obsługiwane rodzaje pamięci: Funkcja obowiązkowa dla wszystkich rodzajów pamięci poza pamięciami SRAM, PSRAM i NOR_FLASH.
6	Verify	MemoryAddr – adres początkowy danych w pamięci zewnętrznej, RAMBufferAddr – adres bufora z danymi źródłowymi, Size – liczba danych (w słowach 16 bit)	0 – operacja zakończona sukcesem ErrAddr – adres pierwszej komórki z błędnymi danymi	Porównanie danych w pamięci zewnętrznej z danymi źródłowymi. Obsługiwane rodzaje pamięci: Funkcja opcjonalna dla wszystkich rodzajów pamięci.



Rysunek 6. Etapy obsługi pamięci X24C08 w programie ST-Link Utility (a) odczyt, b) zapis danymi z pliku, c) prezentacja zawartości pamięci)

Plik *Loader_Src.c* zawiera właściwy kod sterownika pamięci zewnętrznej. Ma on postać 6 funkcji, za pośrednictwem których program *ST-Link Utility* obsługuje pamięć zewnętrzną. Zostały one zebrane w **tabeli 2**. Jak widać, nie wszystkie funkcje muszą być zaimplementowane dla każdego rodzaju pamięci. Część z nich jest albo zbędna, albo opcjonalna. W prezentowanym przykładzie, zarówno dla pamięci X24C08, jak i dla zegara PCF8583, wymagana jest jedynie implementacja funkcji *Init()*, *Read()* i *Write()*. Zostały one przedstawione na **listingu 4**. Dzięki właściwemu doborowi listy argumentów niskopoziomowych funkcji obsługi pamięci EEPROM, implementacja funkcji sterownika ogranicza się w zasadzie do wywołania odpowiedniej funkcji niskiego poziomu obsługującej układ dołączony do magistrali I²C. W przypadku funkcji *Init()* nie było bowiem potrzeby dodatkowej konfiguracji systemu generowania sygnałów zegarowych mikrokontrolera STM32F051. Taktowanie układu z domyślnego generatora HSI o częstotliwości 8 MHz jest bowiem w przypadku obsługi magistrali I²C ze standardową częstotliwością 100 kHz zupełnie wystarczające.

Prezentowany sterownik może zostać dodatkowo rozbudowany o funkcję *Verify()*, dzięki czemu możliwa będzie weryfikacja zawartości pamięci podczas procesu zapisu danych. W omawianym przykładzie zrezygnowano jednak z tej możliwości. W takiej sytuacji zawartość pamięci jest weryfikowana dopiero po zakończeniu zapisu, w drodze odczytu zawartości całego układu.

Instalacja sterownika

Otrzymany w wyniku kompilacji projektu plik wynikowy **.elf* jest w zasadzie gotowym sterownikiem pamięci zewnętrznej. Należy jedynie zmienić rozszerzenie nazwy pliku z *.elf* na *.stldr* i umieścić tak zmodyfikowany plik w folderze *ExternalLoader* programu *ST-Link Utility*. W prezentowanym przykładzie operację tę należy wykonać dwukrotnie, najpierw dla pliku **.elf* uzyskanego przy kompilacji projektu z ustawieniami dyrektyw *#define* i struktury *StorageInfo* dla pamięci EEPROM X24C08, a następnie dla zegara PCF8583. Po uruchomieniu programu *ST-Link Utility* oba układy powinny pojawić się na liście obsługiwanych pamięci zewnętrznych w oknie *Add External Loader* menu *External Loader* (**rysunek 4**). Po zaznaczeniu obu układów i akceptacji dokonanego wyboru program *ST-Link Utility* jest gotowy do obsługi tych układów.

O poprawnym zainstalowaniu w programie *ST-Link Utility* obu sterowników układów dołączonych do magistrali I²C mikrokontrolera świadczy pojawienie się dwóch nowych pozycji w menu *External Loader*. Oba nowe punkty menu odpowiadają odpowiednio pamięci EEPROM X24C08 i zegarowi RTC PCF8583 i pozwalają na odczyt oraz zapis tych układów danymi zawartymi w pliku (**rysunek 5**).

Odczyt i zapis pamięci X24C08 oraz zegara PCF8583 jest także możliwy z poziomu okna głównego programu *ST-Link Utility*. W przypadku wybrania adresu leżącego w zakresie adresów przypisanych pamięci EEPROM (tj. 0x60000000-0x600003FF) zostanie automatycznie załadowany sterownik pamięci EEPROM i odczytana zawartość tej pamięci, natomiast w przypadku wybrania adresu leżącego w zakresie adresów przypisanych zegarowi RTC (tj. 0x60001000-0x600010FF) użyty zostanie sterownik układu PCF8583 i odczytana zawartość tego układu. **Rysunki 6** i **7** przedstawiają różne etapy obsługi układów X24C08 i PCF8583 przez program *ST-Link Utility*.

W przypadku zapisu pierwszych 16 komórek układu PCF8583 program *ST-Link Utility* może sygnalizować błąd zapisu danych. Jest to spowodowane tym, że komórki te są w istocie rejestrami zegara RTC i część z nich, gdy zegar jest uruchomiony, ciągle zmienia swą wartość. Tak dzieje się np. z komórkami o adresach od 0x60001001 do 0x60001006, które są licznikami milisekund, sekund, minut, godzin, dni i miesięcy. W takiej

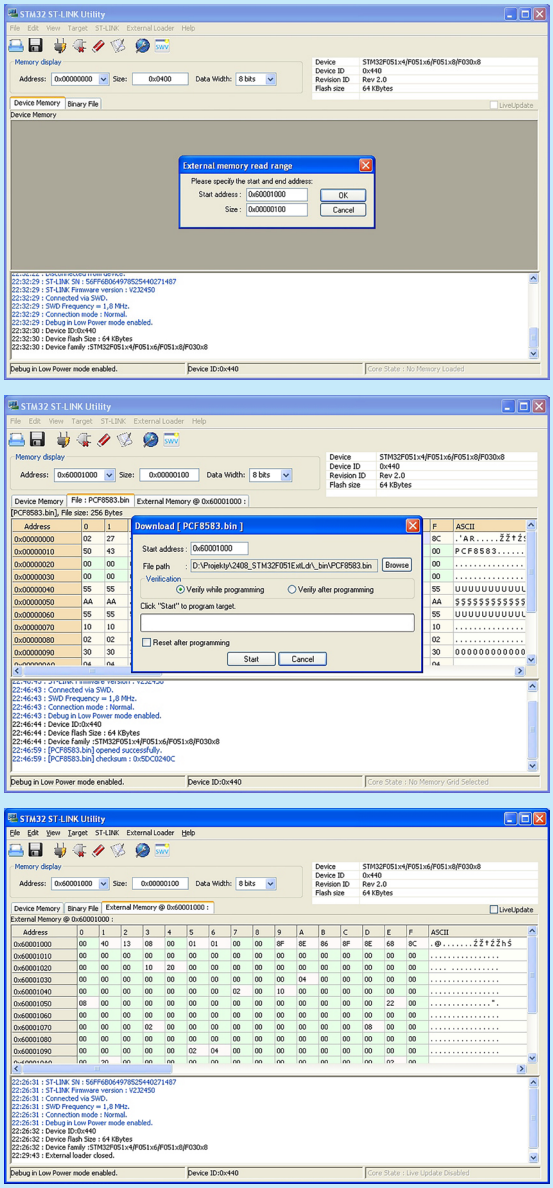
Listing 4: Sterownik pamięci EEPROM X24C08 / zegara PCF8583 (plik *Loader_Src.c*)

```
#include „stm32f0xx.h”
#include „i2c_eeprom.h”
#define StartAddress 0x60000000

int Init (void)
{
    i2c_eeprom_Init();
    return 1;
}

int Read (uint32_t Address, uint32_t Size, uint8_t *Buffer)
{
    return i2c_eeprom_ReadBlock(Buffer, (uint16_t)(Address - StartAddress), Size);
}

int Write (uint32_t Address, uint32_t Size, uint8_t* Buffer)
{
    return i2c_eeprom_WriteBlock(Buffer, (uint16_t)(Address - StartAddress), Size);
}
```



Rysunek 7. Etapy obsługi zegara RTC PCF8583 w programie ST-Link Utility (a) odczyt, b) zapis danymi z pliku, c) prezentacja zawartości pamięci)

sytuacji pomiędzy zapisem danej komórki pamięci przez program *ST-Link Utility* a jej odczytem w celu weryfikacji poprawności zapisu wartość licznika odpowiadającego milisekundom, sekundom, itd. może się zmienić. Stąd sygnalizowany przez program *ST-Link Utility* błąd zapisu danych. Powyższe zachowanie układu PCF8583 można łatwo zaobserwować w programie *ST-Link Utility*, dokonując w regularnych odstępach czasu odczytu zawartości zegara RTC. Komórki odpowiadające licznikom czasu będą zmieniały swoją zawartość zgodnie z upływającym czasem. Sygnalizowany problem błędu zapisu danych nie dotyczy oczywiście komórek pamięci o adresach powyżej 0x60001010, ponieważ są to komórki standardowej pamięci SRAM w układzie PCF8583.

Podsumowanie

Napisanie sterownika pamięci zewnętrznych dla programu *ST-Link Utility* nie jest skomplikowane i nie wymaga wielkich nakładów czasu i pracy, ponieważ zazwyczaj i tak większość funkcji niskiego poziomu obsługujących pamięć zewnętrzną jest tworzona w ramach projektu konstruowanego urządzenia, warto poświęcić kilka dodatkowych chwil i zaimplementować taki sterownik. Może on oddać nieocenione usługi podczas pisania i testowania oprogramowania budowanego układu.

Możliwość odczytu i zapisu w systemie pamięci zewnętrznej dołączonej do mikrokontrolera STM32 rozwiązuje też problem umieszczania w tej pamięci danych startowych, które z takich czy innych powodów nie są przechowywane w wewnętrznej pamięci Flash mikrokontrolera. Mogą to być np. dane identyfikacyjne urządzenia, takie jak jego numer seryjny czy data produkcji lub indywidualne parametry kalibracyjne obwodów pomiarowych, ale nie tylko. Program *ST-Link Utility* może bowiem posłużyć np. do umieszczania w pamięci zewnętrznej zestawów czcionek czy grafik wykorzystywanych przez oprogramowanie urządzenia.

Wreszcie, program *ST-Link Utility* może zostać użyty do testowania układu zewnętrznego, w omawianym przykładzie zegara RTC. Możliwość zapisu i odczytu poszczególnych rejestrów układu z poziomu komputera PC w łatwy sposób pozwala na zapoznanie się z funkcjonowaniem danego układu, bez konieczności umieszczania funkcji testowych w programie mikrokontrolera. Jak widać, możliwości stwarzane przez *ST-Link Utility* są dość szerokie, więc warto wykorzystywać je w pełni.

Aleksander Borysiuk
alex_priv@wp.pl

Bibliografia

1. ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32, User manual UM1075, STMicroelectronics.
2. STM32 ST-LINK Utility software description, User manual UM0892, STMicroelectronics.
3. STM32F0DISCOVERY Discovery kit for STM32F0 microcontrollers, User manual UM1525, STMicroelectronics.
4. 8kbit serial E2PROM X24C08 1024×8 Bit, Product data sheet, Xicor Inc.
5. PCF8583 Clock and calendar with 240×8-bit RAM, Product data sheet, NXP