SC16IS760

UART z interfejsem SPI/I²C

Układy asynchronicznej transmisji szeregowej (UART) należą do standardowego zestawu wewnętrznych układów peryferyjnych większości mikrokontrolerów jednoukładowych. Przeważnie mikrokontrolery są wyposażane w pojedynczy UART, rzadziej jest ich więcej. Zdarzają się również mikrokontrolery pozbawione tego układu. I wówczas, w obliczu braku UART-a, lub konieczności wyposażenia mikrokontrolera w dodatkowe interfejsy, możliwe są dwa rozwiązania: programowe lub zastosowanie układu zewnętrznego.

Najczęściej spotykanym w zastosowaniach UART jest układ scalony 16C450, który jest używany w komputerach PC do realizacji transmisji asynchronicznej. Niestety posiada on równoległe wejście danych oraz najczęściej dostępny jest w 40–wyprowadzeniowej obudowie DIP, co pociąga za sobą znaczny wzrost wymiarów płytki oraz komplikuje połączenia. Trudno też implementować równoległą magistralę do komunikacji z układem UART tylko specjalnie na potrzeby realizacji transmisji szeregowej, zwłaszcza w przypadku braku wolnych wyprowadzeń mikrokontrolera.

Panaceum na opisane problemy są układy SC16IS760 oraz podobne SC16IS740/750. SC16IS760 i SC16IS750 są dostępne w 24–wyprowadzeniowych obudowach TSSOP oraz HVQFN (o wymiarach zaledwie 4×4 mm), natomiast układ SC16IS740 w 16–wyprowadzeniowej obudowie TSSOP. Co ważniejsze, układy te do komunikacji z mikrokontrolerem wykorzystują interfejs SPI lub I²C, dzięki czemu łatwo jest podłączyć je do mikrokontrolera. Ponadto pod względem zestawu rejestrów sterujących są kompatybilne z układem 16C450. Wszystkie wymienione układy posiadają 64–bajtowe bufory FIFO dla nadajnika i odbiornika. Dodatkowo SC16IS750/760 wyposażone są w dwukierunkowy 8–bitowy port GPIO.

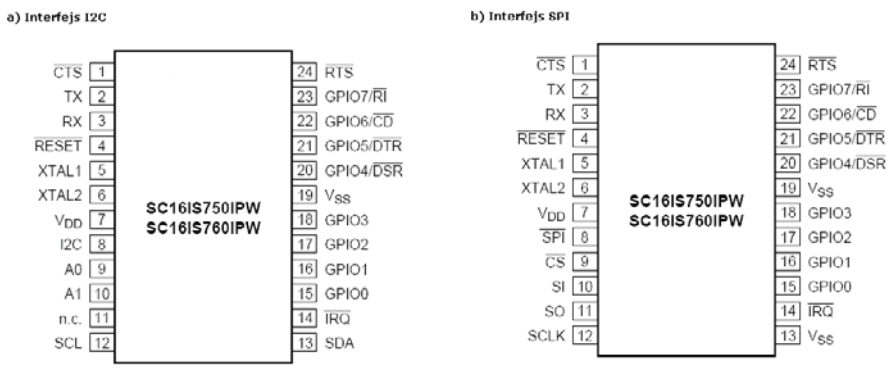
Tab. 1. Opis wyprowadzeń układu SC16IS760

Numer	Oznaczenie	Funkcja
1	CTS	Wyprowadzenie CTS (*Clear To Send*)
2	TX	Wyjście nadajnika UART
3	RX	Wejście odbiornika UART
4	RESET	Wejście zerowania układu
5	XTAL1	Wejście zewnętrznego sygnału zegarowego lub rezonatora kwarcowego
6	XTAL2	Wyjście sygnału zegarowego
7	VDD	Napięcie zasilania
8	I2C/SPI	Wejście wyboru pomiędzy interfejsem I²C a SPI
9	CS/A0	Wybór układu na magistrali SPI albo wyprowadzenie A0 magistrali I²C
10	SI/A1	Wejście danych SPI/wyprowadzenie A1 magistrali I²C
11	SO	Wyjście danych SPI
12	SCL/SCLK	Wejście sygnału SCL interfejsu I²C/wejście sygnału zegarowego SPI
13	SDA	Wyprowadzenie SDA interfejsu I²C.
14	IRQ	Wyjście żądania przerwania
15	GPIO0	Port I/O 0
16	GPIO1	Port I/O 1
17	GPIO2	Port I/O 2
18	GPIO3	Port I/O 3
19	VSS	Masa
20	GPIO4	Port I/O 4
21	GPIO5	Port I/O 5
22	GPIO6	Port I/O 6
23	GPIO7	Port I/O 7
24	RTS	Wyprowadzenie RTS (Ready To Send)

Rozkład wyprowadzeń układu SC16IS760 przedstawiono na rysunku **rys. 1**. W zależności od stanu wyprowadzenia I²C/SPI układ SC16IS760 wykorzystuje do komunikacji interfejs I²C (stan wysoki) lub SPI (stan niski). Opis wyprowadzeń SC16IS760 przedstawiono w **tab. 1**, a zestaw rejestrów w **tab. 2.**

Adresy rejestrów dodatkowych pokrywają się z adresami rejestrów podstawowych i dlatego dostęp do rejestrów DLL oraz DLH jest możliwy wtedy, gdy bit 7 w rejestrze LCR jest równy „1" i jego zawartość jest różna od BFh, natomiast dostęp do rejestrów EFR, XON1, XON2, XOFF1 i XOFF2 uzyskuje się po zapisaniu do rejestru LCR liczby BFh. Rejestry UART wymieniono niżej:

- THR (*Transmit Holding Register*) – rejestr danych do wysłania. Jest to jednocześnie 64-bajtowy bufor FIFO nadajnika. W przypadku, gdy jest wyłączony, nadpisane dane są tracone (stan domyślny po zerowaniu).
- RHR (*Receive Holding Register*) – rejestr odebranych danych. Rejestr ten również jest 64–bajtowym buforem FIFO.
- FCR (*FIFO Control Register*) – kontroluje pracę buforów FIFO. Funkcje poszczególnych bitów rejestru FCR zestawiono w **tab. 3**.
- LCR (*Line Control Register*) – określa format transmitowanych danych (**tab. 4**).
- LSR (*Line Status Register*) – rejestr statusu transmisji danych (**tab. 5**).



Rys. 1. Rozmieszczenie wyprowadzeń układów UART

– IODir (*I/O Direction Register*) – wybór kierunku pracy wyprowadzeń I/O ogólnego przeznaczenia. Wyprowadzenie pracuje jako wejście, gdy odpowiadający mu bit w rejestrze IODir jest wyzerowany.
– IOState (*I/O State Register*) – stan wyprowadzeń I/O. Wyzerowanie/ustawienie bitu rejestru powoduje tę samą reakcję odpowiedniego wyprowadzenia portu. Odczyt rejestru zwraca stan panujący na wyprowadzeniach.
– IOIntEna (*I/O Interrupt Enable Register*) – zezwolenie na przerwanie od zmiany stanu wyprowadzenia wejściowego. Ustawienie bitu w rejestrze spowoduje zezwolenie na przerwanie.
– IOControl (*I/O Control Register*) – rejestr służy do konfiguracji portu GPIO. Składa się z bitów, których przeznaczenie zostało omówione w **tab. 6**.

Komunikacja za pośrednictwem interfejsu SPI – przykłady dla mikrokontrolerów ST7LITE

Mikrokontrolery ST7LITE, poza ST7LITE3x, nie posiadają układu UART. W dalszej części artykułu zostanie omówiona współpraca układu SC16IS760 z komputerkiem LITEcomp wyposażonym w mikrokontroler ST7FLITE19. Schemat połączeń pomiędzy układem SC16IS760 a LITEcompem przedstawiono na **rys. 2**. Komunikacja z układem SC16IS760 zostanie zrealizowana poprzez interfejs SPI, a wyprowadzenie PB0 będzie pełnić rolę sygnału CS. Stan niski CS powoduje załączenie interfejsu SPI układu SC16IS760.

Procedury obsługi interfejsu SPI. Realizacja transmisji z wykorzystaniem interfejsu SPI jest stosunkowo prosta ze względu na fakt wyposażenia mikrokontrolera ST7FLITE19 w sprzętowy układ SPI. Obsługa wyprowadzenia PB0, pełniącego funkcję sygnału CS dla układu SC16IS760 jest zrealizowana programowo. Procedury obsługi transmisji SPI znajdują się w pliku *SPI.asm*, który należy kompilować wspólnie z *SC16IS760.asm*.

Procedura inicjalizacji interfejsu SPI. Interfejs SPI musi być odpowiednio skonfigurowany. Służy do tego procedura `SPI_Init`. Na samym początku konfigurowane jest wyprowadzenia pełniące funkcję sygnału uaktywniającego interfejs SPI w układzie SC16IS760. Wyprowadzenie PB0, przypisane do nazwy symbolicznej `SPI_CS` za pomocą dyrektywy `#define SPI_CS PB0`, jest konfigurowane jako wyjście oraz ustawiane w stan wysoki. Interfejs SPI musi oczywiście pracować w trybie Master, toteż ustawiany jest bit MSTR w rejestrze SPICR. Dodatkowo należy wyłączyć alternatywną funkcję wyprowadzenia PB0 (SS). W tym celu należy ustawić bity SSM oraz SSI, które znajdują się w rejestrze SPISR. Częstotliwość sygnału SCK ustawiono na wartość FCPU/8. Odpowiadają za to bity SPR2... SPR0 w rejestrze SPICR. W przypadku pracy mikrokontrolera ST7FLITE19 z zewnętrznym rezonatorem kwarcowym o częstotliwości 16 MHz, częstotliwość sygnału SCK wyniesie 1 MHz, ponieważ wewnętrznie ST7LITE dzieli częstotliwość zegarową przez 2.

Tab. 2. Rejestry układu SC16IS760

Adres	Nazwa	Funkcja	Kierunek
Rejestry podstawowe			
0x00	RHR	Receive Holding Register	R
0x00	THR	Transmit Holding Register	W
0x01	IER	Interrupt Enable Register	R/W
0x02	FCR	FIFO Control Register	W
0x02	IIR	Interrupt Identification Register	R
0x03	LCR	Line Control Register	R/W
0x04	MCR	Modem Control Register	R/W
0x05	LSR	Line Status Register	R
0x06	MSR	Modem Status Register	R
0x07	SPR	Scratchpad Register	R/W
0x06	TCR	Transmission Control Register	R/W
0x07	TLR	Trigger Level Register	R/W
0x08	TXLVL	Transmit FIFO Level Register	R
0x09	RXLVL	Receive FIFO Level Register	R
0x0A	IODir	I/O pin Direction Register	R/W
0x0B	IOState	I/O pin States Register	R/W
0x0C	IOIntEna	I/O Interrupt Enable Register	R/W
0x0D	-	-	-
0x0E	IOControl	I/O Control Register	R/W
0x0F	EFCR	Extra Features Register	R/W
Rejestry dodatkowe			
0x00	DLL	Divisor Latch LSB	R/W
0x01	DLH	Divisor Latch MSB	R/W
0x02	EFR	Enhanced Feature Register	R/W
0x04	XON1	Xon1 word	R/W
0x05	XON2	Xon2 word	R/W
0x06	XOFF1	Xoff1 word	R/W
0x07	XOFF2	Xoff2 word	R/W

Tab. 3. Funkcje bitów rejestru FCR

Bit	Funkcja	Opis
7:6	RX Trigger	Liczba znaków w buforze odbiornika generująca przerwanie: 00=8, 01=16, 10=56, 11=60 znaków
5:4	TX Trigger	Liczba wolnych komórek pozostających w buforze nadajnika generująca przerwanie: 00=8, 01=16, 10=32, 11=56
3	zarezerwowany	
2	Reset TX FIFO	Ustawienie powoduje zerowanie bufora FIFO nadajnika
1	Reset RX FIFO	Ustawienie powoduje zerowanie bufora FIFO odbiornika
0	FIFO Enable	Ustawienie powoduje załączenie buforów FIFO nadajnika i odbiornika

Tab. 4. Funkcje bitów rejestru LCR

Bit	Funkcja	Opis
7	Divisor Latch Enable	Ustawienie umożliwia na dostęp do rejestrów DLL i DLH
6	Break control	Ustawienie powoduje wymuszenie na linii TX stanu niskiego (sygnał BREAK)
5	Set parity	Bit parzystości, gdy bit Parity enable=1
4	Parity type	0=bit parzystości ustawiony przy parzystej liczbie 1,1=bit parzystości ustawiony przy nieparzystej liczbie 1
3	Parity enable	0=brak bitu parzystości, 1=bit parzystości jest generowany przez nadajnik i sprawdzany przez odbiornik
2	Stop bits	Bit określa ilość bitów stopu 0=1 bit stopu, 1=1,5 bitu stopu dla 5-bitowego słowa danych, 1=2 bity stopu dla słowa danych 6...8 bitów
1:0	Word length	Długość słowa danych: 00=5, 01=6, 10=7, 11=8 bitów

Procedura transmisji bajtu danych. Do wysłania bajtu danych służy procedura `SPI_Send`. Transmisja jest inicjowana przez zapis do rejestru SPDR bajtu przeznaczonego do wysłania. Następnie sprawdzany jest stan flagi SPIF w rejestrze SPISR, która jest ustawiana po zakończeniu transmisji bajtu. Po przesłaniu bajtu w rejestrze SPDR znajdzie się odebrany bajt danych, który jest kopiowany do rejestru A.

Tab. 5. Funkcje bitów rejestru LSR

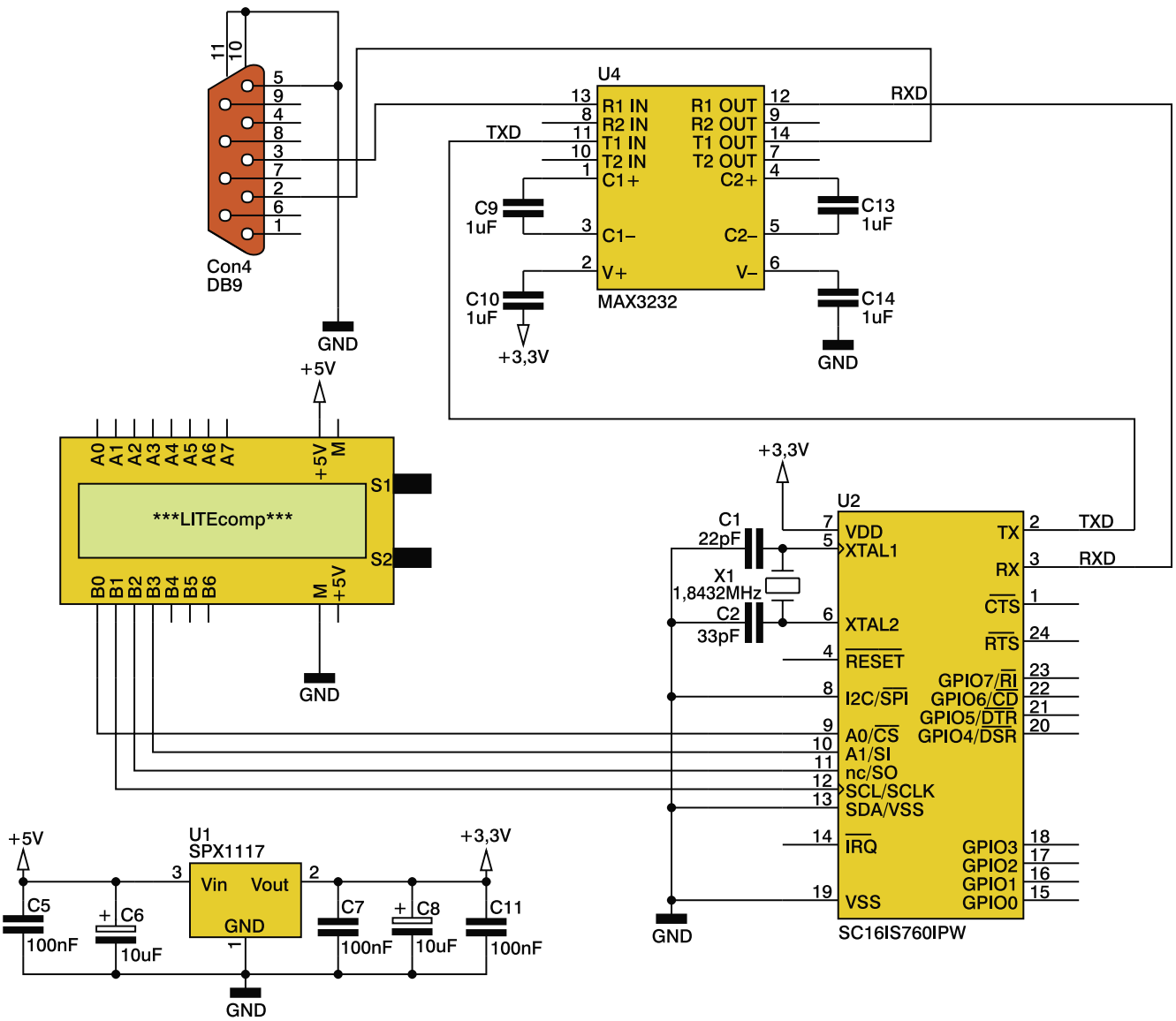
Bit	Funkcja	Opis
7	FIFO data error	Ustawiony bit sygnalizuje co najmniej jeden błąd parzystości, formatu ramki danych znajdujących się w buforze FIFO odbiornika. Bit jest zerowany, gdy dane przy odbiorze których wystąpił błąd opuszczą bufor FIFO
6	THR and TSR empty	Ustawiony bit sygnalizuje, że rejestry THR oraz TSR są puste.
5	THR empty	Ustawiony bit sygnalizuje, że rejestr THR jest pusty.
4	Break	Ustawiony bit sygnalizuje odebranie sygnału BREAK.
3	Framing error	Ustawiony bit sygnalizuje błąd ramki danych
2	Parity error	Ustawiony bit sygnalizuje błąd parzystości danych
1	Overrun error	Ustawiony bit sygnalizuje przepełnienie bufora odbiornika
0	Data in receiver	Ustawiony bit sygnalizuje obecność co najmniej jednego znaku w buforze FIFO odbiornika

Tab. 6. Nastawy bitów rejestru IOControl

Bit	Funkcja	Opis
7:4	-	Zarezerwowane
3	SRESET	Programowe zerowanie układu. Zapis jedynki do tego bitu spowoduje restart układu
2	-	Zarezerwowany
1	GPIO lub sygnały Modemu	0=wyprowadzenia GPIO[7...4] jako porty I/O ogólnego przeznaczenia 1=wyprowadzenia GPIO[7...4] jako RI, CD, DTR, DSR.
0	IOLATCH	Stan wejść zapamiętywany (1) lub nie (0) po wystąpieniu przerwania IOState.

Procedury obsługi linii CS. Kod zarządzający stanem linii CS został wydzielony do samodzielnej procedury o nazwie `SPI_CS_Enable`. W przypadku, gdy wykorzystujemy tylko jeden układ SC16IS760, a co za tym idzie pojedynczy port I/O, stworzenie z jednej instrukcji asemblera procedury wydaje się być rozrzutnością, jednak ma to sens w sytuacji, gdy w projekcie zechcemy wykorzystać większą liczbę układów CS16IS760, lub też inne układy pracujące na magistrali SPI. Można dokonać takiej modyfikacji procedur zarządzania sygnałami CS, aby wybór aktywnego sygnału był dokonywany poprzez przekazanie argumentu przez rejestr Y. W ten sposób kod będzie bardziej uniwersalny i w prosty sposób można go przystosować, o ile zajdzie taka potrzeba, do współpracy z większą liczbą układów pracujących na magistrali SPI.

Zapis danej do rejestru. Przebiegi na magistrali SPI podczas zapisu do rejestru wewnętrznego układu SC16IS760 przedstawiono na **rys. 3**. Procedura zapisu do rejestru (`SC16IS760_WriteReg`) sprowadza się do uaktywnienia sygnału CS oraz zapisu dwóch bajtów:



Rys. 2. Schemat połączenia układu SC16IS760IPW z płytką LITEcomp

List. 1. Program do transmisji danych przez UART podłączony do interfejsu SPI

```
st7/
;--------------------------------------------------------------------------------
; SC16IS760 - Single UART with I2C/SPI and 64 bytes FIFO
; współpraca z mikrokontrolerem ST7FLITE19 (SPI)
; radoslaw.kwiecien@ep.com.pl
;--------------------------------------------------------------------------------
.nolist
#INCLUDE „st7flite19.inc”
  #INCLUDE „HD44780.inc”
  #INCLUDE „SC16IS760.inc”
  #INCLUDE „SPI.inc”
.list
;--------------------------------------------------------------------------------
; obszar pamięci danych RAM
;--------------------------------------------------------------------------------
  BYTES
  segment ‚ram0’
;--------------------------------------------------------------------------------
; obszar pamięci programu
;--------------------------------------------------------------------------------
  WORDS
  segment ‚rom’
napis1 STRING „SC16IS760 - Single UART with I2C/SPI and 64 bytes FIFO-
”,10,13,0
napis2 STRING „Elektronika Praktyczna - www.ep.com.pl”,10,13,0
napis3 STRING „ --SC16IS760--”,0

;--------------------------------------------------------------------------------
; początek programu głównego
;--------------------------------------------------------------------------------
._ProgramEntryPoint
  CALL LCD_Initialize       ; inicjalizacja LCD
  CALL SPI_Init             ; inicjalizacja SPI
  LD A,#napis3.h
  LD {textPointer+0}, A
  LD A,#napis3.l
  LD {textPointer+1}, A
  CALL LCD_WriteText        ; wyświetlenie napisu na LCD
  LD A,#SECLINEADDR
  CALL LCD_SetAddress       ; przejście do drugiej linii LCD
  LD Y,#5

LD X,#LCR
  LD A,#DIVISORLATCHENABLE
  CALL SC16IS760_WriteReg

LD X,#DLL
  LD A,#BR_9600
  CALL SC16IS760_WriteReg


LD X,#DLH
  LD A,#0
  CALL SC16IS760_WriteReg

LD X,#LCR
  LD A,#{DATABITS8+ONESTOP+NOPARITY}
  CALL SC16IS760_WriteReg

LD X,#FCR
  LD A,#{FIFOENABLE+RESETRXFIFO+RESETTXFIFO}
  CALL SC16IS760_WriteReg

LD X,#napis1.l
  LD A,#napis1.h
  CALL SC16IS760_WriteStr   ; zapis napisu do bufora FIFo
WaitTXEmpty
  LD X, #LSR
  CALL SC16IS760_ReadReg
  AND  A,#%00100000
  JREQ WaitTXEmpty        ; oczekiwanie na wysłanie danych
  LD X,#napis2.l
  LD A,#napis2.h
  CALL SC16IS760_WriteStr   ; zapis napisu do bufora FIFo
MainLoop
  LD Y,#5
  LD X,#LSR
  CALL SC16IS760_ReadReg
  AND  A,#1
  JREQ MainLoop        ; oczekiwanie na odebranie znaku
  LD X,#RHR
  CALL SC16IS760_ReadReg    ; odczyt odebranego znaku
  CALL LCD_WriteData     ; wyświetlenie go na LCD
  JRA  MainLoop
  END
```

adresu wewnętrznego rejestru oraz bajtu zawierającego zapisywaną do rejestru wartość.

Bit 7 określa, czy mamy do czynienia z zapisem (0), czy odczytem rejestru (1), natomiast adres rejestru wewnętrznego jest odwzorowany na bitach 6...3. W związku z tym, że do rejestrów zapisywane są dane konfiguracyjne, adres rejestru należy przesunąć o 3 bity w prawo, natomiast pozostałe bity powinny mieć wartość logicznego zera.

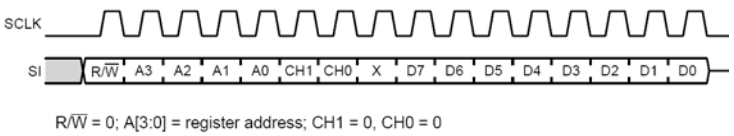
Przed wywołaniem procedury do rejestru X należy załadować adres rejestru wewnętrznego układu SC16IS760, natomiast do rejestru A mikrokontrolera wartość przeznaczoną do zapisania w rejestrze SC16IS760.

Zapis łańcucha znaków do bufora nadawczego. Oprócz możliwości zapisu pojedynczego bajtu do rejestru wewnętrznego układu SC16IS760 istnieje możliwość ciągłego zapisu większej ilości danych do bufora nadajnika. Dzięki temu, przy transmisji większej ilości danych, zmniejsza się obciążenie jednostki centralnej mikrokontrolera. Ciągła transmisja wymaga włączenia bufora FIFO. Naturalnie nie wolno jednorazowo zapisywać danych w ilości większej, niż maksymalna, ustawiona wielkość bufora FIFO. Większe bloki należy podzielić na pakiety o rozmiarze nieprzekraczającym rozmiaru bufora. Pomiędzy zapisem poszczególnych pakietów należy sprawdzić, czy zapisane uprzednio dane zostały wysłane. Procedura przesyłająca łańcuch znaków nosi nazwę `SC16IS760_WriteStr`.

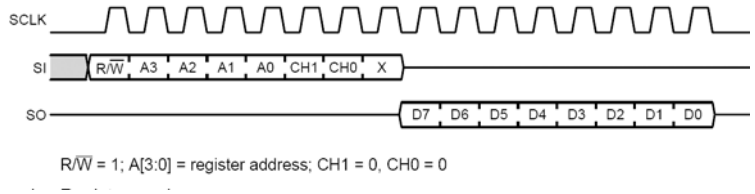
Odczyt bajtu z rejestru. Przebiegi na magistrali SPI podczas odczytu rejestru przedstawione są na **rys. 4.** Procedura odczytująca dane nosi nazwę `SC16IS760_ReadReg`. Podobnie jak w przypadku zapisu, pierwszy transmitowany bajt zawiera adres rejestru. Dodatkowo najstarszy bit tego bajtu jest ustawiony – sygnalizowana jest w ten sposób operacja odczytu.

Przykładowy program wykorzystujący poznane procedury. W celu zobrazowania wykorzystania przedstawionych powyżej procedur przygotowałem prosty program demonstrujący ich zastosowanie. Program ten będzie wyświetlał na wyświetlaczu LCD modułu LITEcomp odbierane za pośrednictwem układu SC16IS760 znaki nadawane przez program terminalowy uruchomiony na komputerze PC. Dodatkowo, po uruchomieniu programu zostaną wysłane dwa napisy. Wykorzystane zostały tu wszystkie omówione wcześniej procedury. Współpraca mikrokontrolera ST7LITE19 z wyświetlaczem LCD była omawiana w ramach kursu „LITEcomp – aplikacje” publikowanego w Elektronice Praktycznej w roku 2007. Zasadniczy kod programu umieszczony w pliku *main.asm* przedstawiono na **list. 1**.

Kompletny projekt dla środowiska ST Visual Develop zawierający przedstawiony program oraz wszystkie niezbędne procedury jest dołączony do artykułu. Komplementarna wersja programu wykorzystująca do komunikacji z układem SC16IS760 interfejs I²C również została dołączona do artykułu.



Rys. 3. Przebiegi na magistrali SPI podczas zapisu do rejestru wewnętrznego



Rys. 4. Przebiegi na magistrali SPI podczas odczytu rejestru wewnętrznego

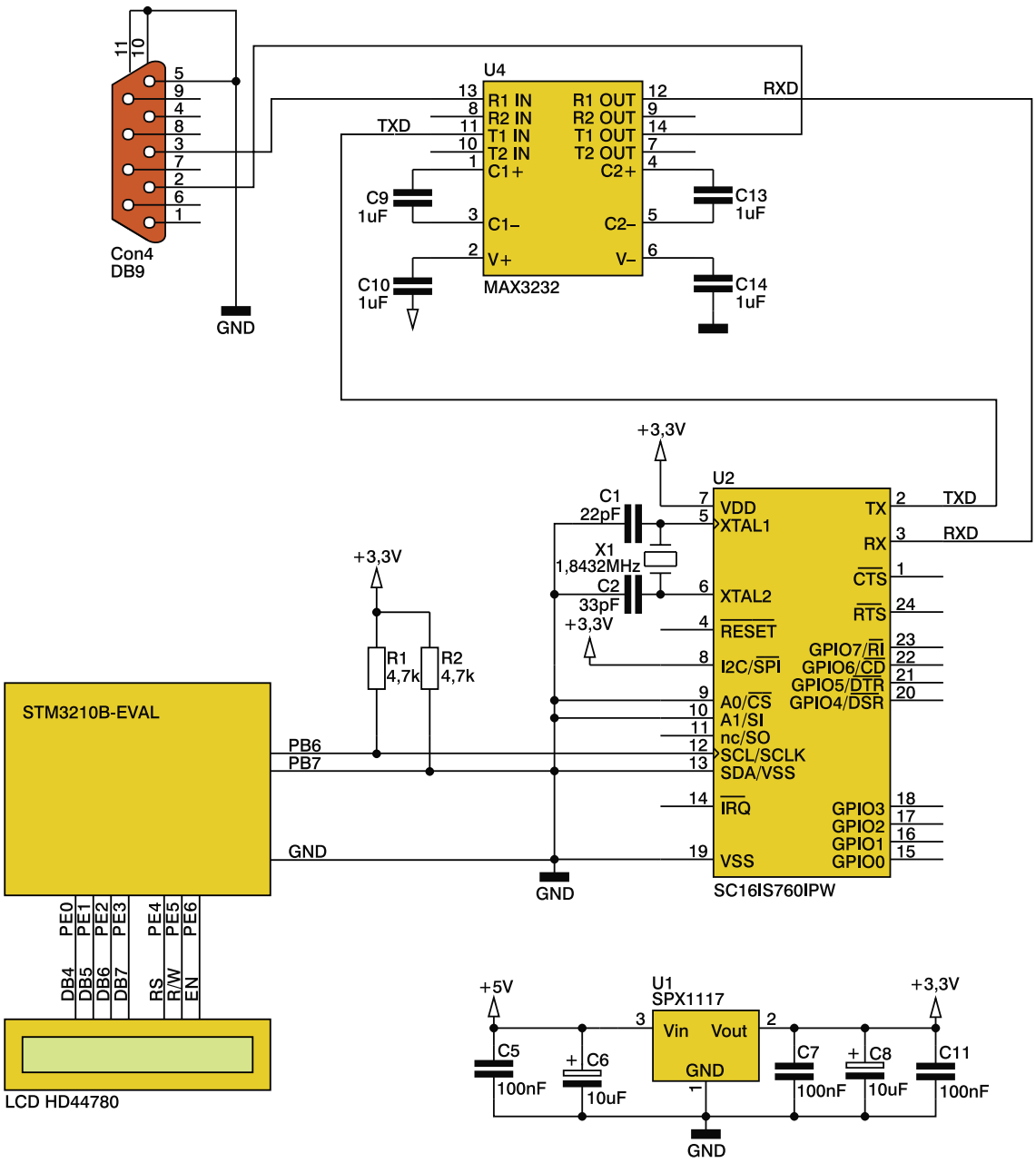
Komunikacja za pośrednictwem interfejsu I²C – przykłady dla mikrokontrolerów STM32

Do pokazania przykładowej komunikacji z układem SC16IS760 poprzez interfejs I²C zostanie użyty mikrokontroler STM32F103VBT6. Wykorzystano do tego bibliotekę STM32 Firmware Library, która pozwala na szybkie i proste pisanie aplikacji w języku C dla mikrokontrolerów STM32. Mikrokontroler STM32F103VBT6 ma dwa sprzętowe interfejsy I²C. Do komunikacji z układem SC16IS760 zostanie wykorzystany interfejs I2C1. Rolę wyprowadzenia sygnału SDA pełni wyprowadzenie PB7, natomiast sygnału SCL wyprowadzenie PB6.

Schemat podłączenia układu SC16IS760 pracującego w trybie I²C przedstawiono na **rys. 5.** Linie A0 i A1 służą do ustalenia adresu układu. Może on przyjąć 16 różnych wartości, tak więc na jednej magistrali I²C może pracować aż 16 układów SC16IS760.

Zapis do rejestru. Przebiegi na magistrali I²C podczas zapisu do rejestru przedstawiono na **rys. 6**. Transmisja rozpoczyna się od sygnału START. Następnie przesyłany jest bajt adresu układu z wyzerowanym bitem R/W. Po otrzymaniu potwierdzenia, transmitowany jest bajt zawierający adres rejestru.

Słowo danych przesyłane do układu ma identyczny format, jak w przypadku SPI. Adres rejestru znajduje się na pozycji bitów 6...3, na-



Rys. 5. Schemat połączenia układu SC16IS760IPW z płytką STM3210B-EVAL

tomiast pozostałe bity powinny być wyzerowane. Ponieważ adres rejestru przekazany jako argument funkcji jest zapisany na czterech najmłodszych bitach, przed wysłaniem do układu SC16IS760 należy go przesunąć o 3 bity w lewo.

Funkcja wysyłająca dane nosi nazwę `void SC16IS760_WriteReg(u8 AddressIndex, u8 RegAddress, u8 dataToWrite)`. Jak widać wymaga ona trzech argumentów. Pierwszym z nich jest indeks tablicy z adresami układu SC16IS760, który może przyjmować wartości od 0 do 15. Drugim argumentem jest adres rejestru, do którego ma zostać zapisany trzeci argument funkcji, jakim jest zapisywany bajt danych.

Odczyt z rejestru. Przebiegi na magistrali I²C podczas odczytu wartości z rejestru przedstawiono na **rys. 7**. Transmisja rozpoczynana się od wysłania sygnału START, adresu układu z wyzerowanym bitem R/W oraz adresu odczytywanego rejestru. Następnie generowany jest ponownie sygnał START oraz transmitowany jest adres układu SC16IS760 z ustawionym bitem R/W sygnalizującym odczyt. W tym momencie dane są nadawane przez układ SC16IS760, natomiast mikrokontroler je odbiera i potwierdza sygnałem ACK w przypadku odczytu więcej niż jednego bajtu danych lub w przypadku odbioru jedynego lub ostatniego bajtu danych, mikrokontroler wysyła sygnał NACK. Odczyt większej ilości danych ma sens jedynie w przypadku odczytu bufora FIFO odbiornika.

Funkcję odbierającą dane zadeklarowano jako `u8 SC16IS760_ReadReg(u8 AddressIndex, u8 RegAddress)`. Przyjmuje ona dwa argumenty: indeks adresu układu SC16IS760 oraz adres rejestru, który będzie odczytywany. Zwraca wartość odczytanego rejestru.

Zapis łańcucha znaków do bufora nadawczego. Funkcja `void SC16IS760_WriteStr(u8 AddressIndex, char * pStr)` zapisuje do bufora FIFO nadajnika ciąg znaków zakończony zerem. Funkcja przyjmuje dwa parametry: indeks adresu układu SC16IS760 na magistrali I²C oraz wskaźnik do tablicy znaków zakończonej zerem.

Funkcja nie dokonuje sprawdzenia, czy w buforze FIFO jest wolne miejsce. Należy więc przed wywołaniem funkcji sprawdzić, czy w buforze FIFO znajduje się odpowiednia ilość wolnych komórek pamięci.

Przykładowy program. Program będący przykładem wykorzystania przedstawionych w artykule funkcji realizuje podobne zadanie, jak program dla mikrokontrolera ST7. Po włączeniu zasilania następuje konfiguracja układu SC16IS760, następnie jest wysyłany przykładowy napis. W pętli nieskończonej program oczekuje na odebranie znaku, który zostanie wyświetlony na wyświetlaczu LCD dołączonym do mikrokontrolera. Kod programu przedstawiono na **list. 2.**

Do artykułu dołączone są (CD EP1/2009B) dwa kompletne projekty dla środowiska Ride7: jeden zawiera omówione w artykule funkcje i przykładowy program do komunikacji z układem SC16IS760 za pośrednictwem magistrali I²C, natomiast drugi został przygotowany do komunikacji za pośrednictwem interfejsu SPI.

Radosław Kwiecień, EP
radoslaw.kwiecien@ep.com.pl

List. 2. Program do transmisji danych przez UART podłączony do interfejsu I²C dla mikrokontrolera STM32

```
//-----------------------------------------------------------------------------
// SC16IS760 - Single UART with I2C/SPI and 64 bytes FIFO
// współpraca z mikrokontrolerem STM32F103VBT6(I2C)
// radoslaw.kwiecien@ep.com.pl
//-----------------------------------------------------------------------------
#include „stm32f10x_lib.h”
#include „SC16IS760.h”

int main(void)
{
SystemInit();

LCD_Initialize();
LCD_WriteTextXY(„SC16IS760”, 0, 0);
LCD_GoTo(0,1);

SC16IS760_WriteReg(5, LCR, DIVISORLATCHENABLE);
SC16IS760_WriteReg(5, DLL, BRL_38400);
SC16IS760_WriteReg(5, DLH, BRH_38400);
SC16IS760_WriteReg(5, LCR, DATABITS8|ONESTOP|NOPARITY);
SC16IS760_WriteReg(5, FCR, FIFOENABLE|RESETRXFIFO|RESETTXFIFO);

SC16IS760_WriteStr(5, „SC16IS760 - Single UART with I2C/SPI and 64 bytes FIFO\n\r”);

do{
  while((SC16IS760_ReadReg(5, LSR)& 0x01) == 0);
  LCD_WriteData(SC16IS760_ReadReg(5, RHR));
}
while(1);
return 0;
}
//-----------------------------------------------------------------------------
// Koniec pliku main.c
//-----------------------------------------------------------------------------
```

Rys. 6. Przebiegi na magistrali I²C podczas zapisu do rejestru

Rys. 7. Przebiegi na magistrali I²C podczas odczytu rejestru