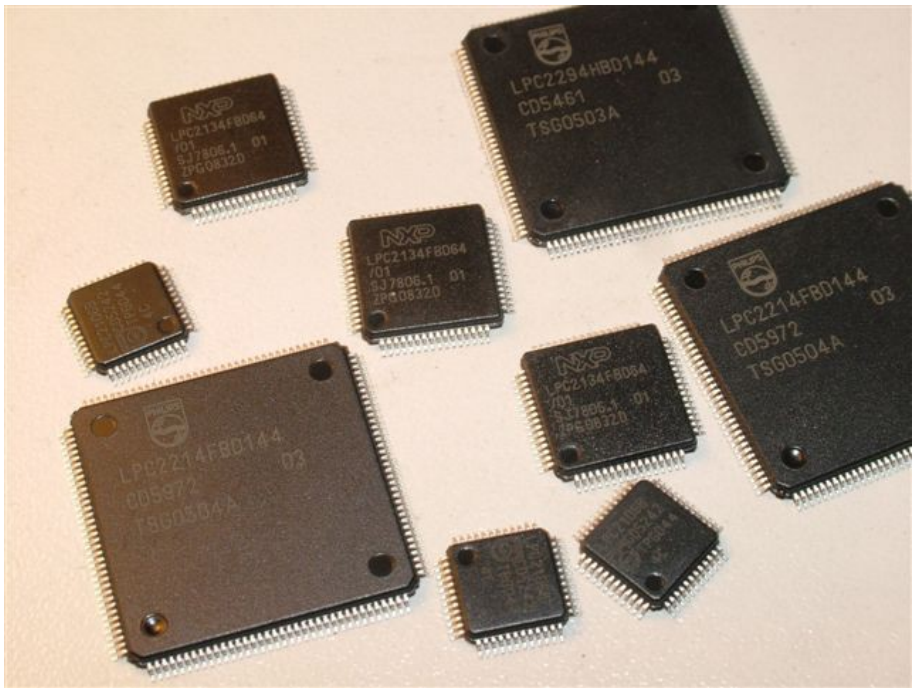
Andrzej Pawluczuk

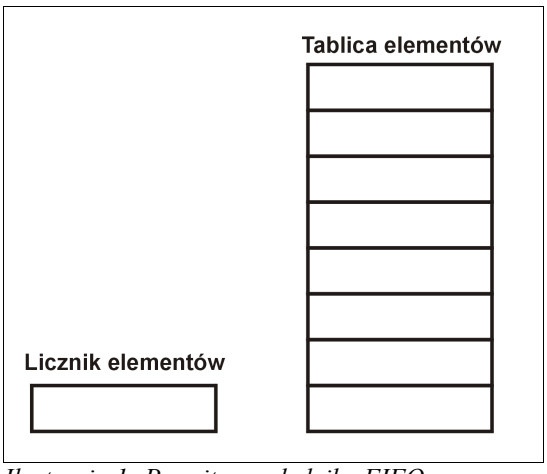


Bufory cykliczne i ich zastosowanie do organizacji kolejek typu FIFO

dla Elportal.pl

Jednym z częściej występujących zagadnień w programach jest ogólnie pojęta problematyka związana z kolejkowaniem informacji. Pod pojęciem kolejkowania należy rozumieć czynność zbierania i później dystrybuowania tych danych (lub szeroko pojętych informacji) z określoną kolejnością. Jednym z częściej występujących jest kolejka FIFO (od ang. **F**irst **I**n **F**irst **O**ut – pierwsze weszło pierwsze wyszło). Można to sobie wyobrazić jako klasyczną kolejkę (taką jak przykładowo występuje w sklepach). Każde elementy dołączające się do kolejki z czasem zostają z niej usunięte (kolejka posuwa się do przodu). Bardzo istotną cechą tu występującą jest kolejność z jaką elementy kolejki do niej wchodzą oraz wychodzą – jest identyczna. Oczywiście jest wiele rozwiązań organizacji i zarządzania strukturą danych emulująca kolejkę FIFO. Stwierdzenie, że jest tyle rozwiązań, co konstruktorów jest może pewną przesadą, ale można zaproponować przynajmniej dwa. Tak jak w wielu przypadkach, każde rozwiązanie ma swoje wady i zalety.

W każdym jednak przypadku musi istnieć coś, co jest w stanie przechowywać elementy kolejki. W przypadku programów dla mikrokontrolerów jest to jakaś struktura danych zapisana w pamięci RAM. Samo „ciało" kolejki jest implementowane jako tablica elementów (każdy język programowania dysponuje „wynalazkiem" określanym jako ARRAY, chociaż przykładowo w języku C nie ma słowa takiego kluczowego, ale jest taka struktura danych). W najprostszym przypadku pojedynczym elementem tablicy jest bajt (interpretowany jako znak lub liczba 8-bitowa). W takim przypadku kolejka FIFO będzie przechowywać ciąg znaków. Oczywiście budowa pojedynczego elementu może być bardziej złożona, czyli zawierać w sobie jakąś strukturę (struktura jako pewna reprezentacja formatu danych występująca w języku C). Jeżeli obok samej tablicy będzie dodatkowa informacja, która będzie reprezentować liczbę elementów, to taki „wynalazek" będzie spełniać funkcję kolejki FIFO (ilustracja 1).

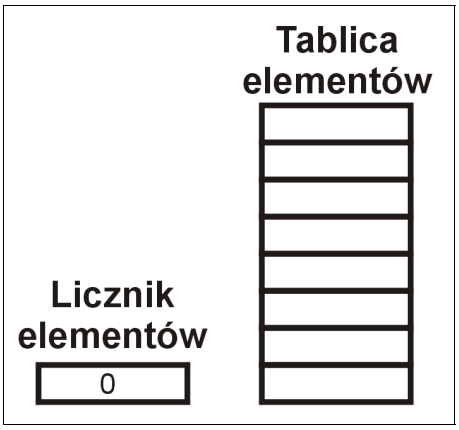


Ilustracja 1: Prymitywna kolejka FIFO

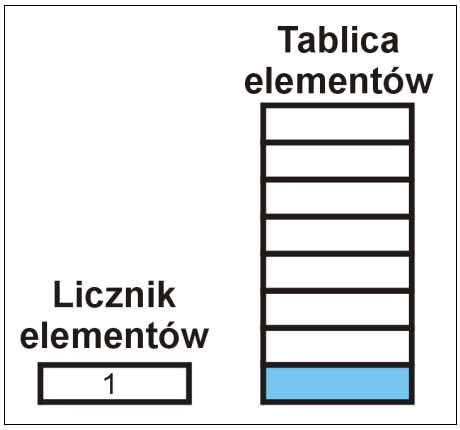
Występują to proste operacje związane z dołożeniem kolejnego elementu: należy zapisać w wolnym miejscu dodawany element i zwiększyć zapisaną liczbę elementów o jeden.

Operacje dodawania elementów pokazują ilustracje 2, 3, 4 i 5. Również liczba elementów jest informacją prostą do uzyskania (by nie powiedzieć, że jest dana wprost). Znacząco gorzej wygląda operacja pobrania elementu z kolejki. Wiadomo, że należy pobrać pierwszy włożony, czyli znajdujący się najniżej, ale to wymaga przemieszczenia wszystkich niepobranych elementów o jedną pozycję w dół, a to może wiązać się czasami ze sporymi kosztami (w sensie wiele operacji wykonywanych z przepisywaniem obszarów danych), ponieważ element, który ma zostać pobrany zawsze znajduje się najniżej. O ile można czasami to „wybaczyć", o tyle w programach mogą zaistnieć

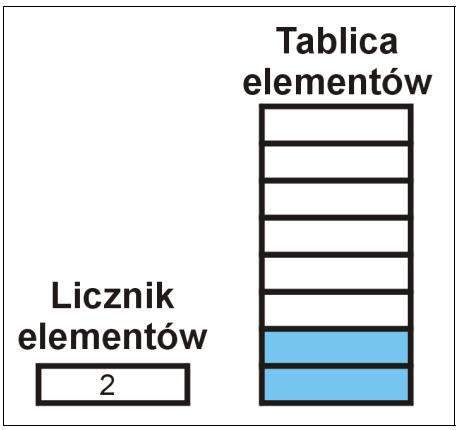
czasami sekcje krytyczne, czyli takie, które muszą być wykonane maksymalnie szybko (jak choćby obsługa przerwania, w której jest realizowana operacja pobierania elementów z takiej kolejki).



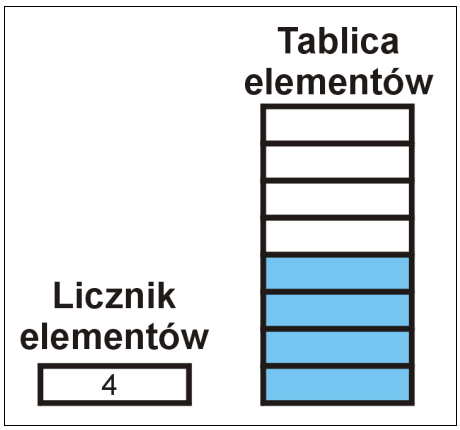
Ilustracja 2: Kolejka pusta



Ilustracja 3: Kolejka z jednym elementem

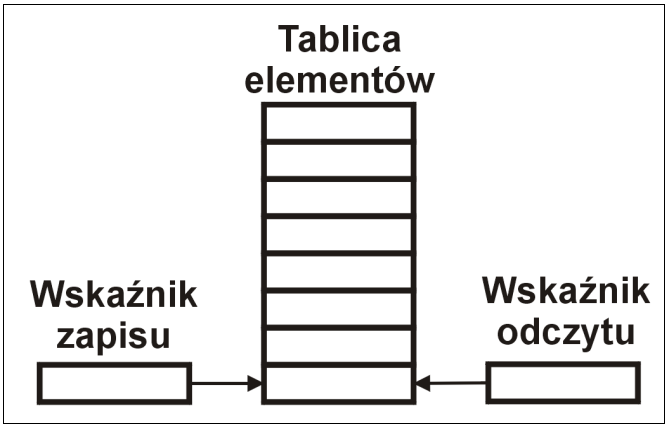


Ilustracja 4: Kolejka z dwoma elementami



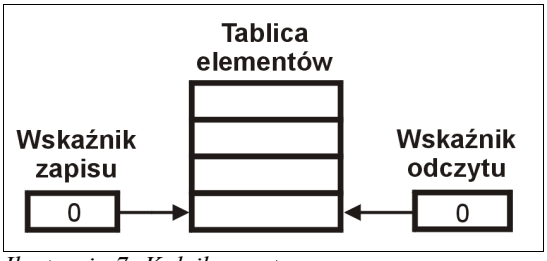
Ilustracja 5: Kolejka z czterema elementami

Można zastąpić element pełniący funkcję licznika elementów zapisanych w tablicy dwoma wskaźnikami, które będą pokazywać miejsce pierwsze wolne (dla operacji zapisu danych) oraz miejsce „czoła kolejki" (dla operacji odczytu). Proponowana struktura może wyglądać jak pokazuje ilustracja 6. Wskaźnik zapisu pokazuje pierwsze wolne miejsce w tabeli, wskaźnik odczytu miejsce czoła kolejki (pierwsze do odczytu). Oczywiście w tej sytuacji komplikuje się określenie liczby elementów zawartych z całej kolejce. Operacja zapisu sprowadza się w tym układzie do zapisu kolejkowanych

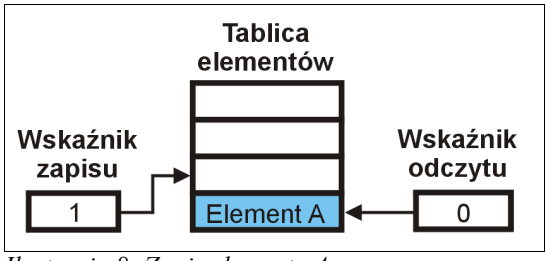


Ilustracja 6: Tablica z dwoma wskaźnikami

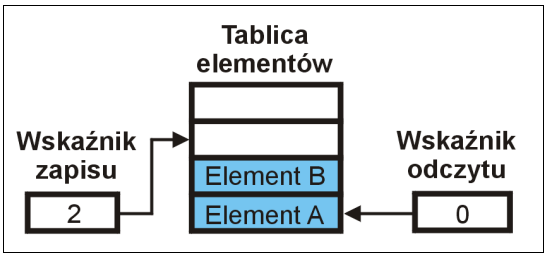
danych do tablicy w miejsce wskazane przez wskaźnik zapisu (który należy interpretować jako indeks do tablicy), zwiększenie wskaźnika o jeden. Oczywiście po pewnym czasie ciągle zwiększając wskaźnik „wyjdzie on poza tablicę", więc w takiej chwili należy go zawrócić na pozycję pierwszą (co w przypadku języka programowania C oznacza, że należy wskaźnik wyzerować, gdyż tablice zawsze są indeksowane od zera). Z kolei operacja odczytu wiąże się z pobraniem elementu z miejsca wskazanego przez wskaźnik odczytu (również traktowanego jako indeks do tablicy), zwiększenia wskaźnika o jeden z ewentualnym „zawróceniem" wskaźnika na początek tablicy w sytuacji, gdy wskaźnik poprzednio wskazywał element ostatni w tablicy. Operacja pobrania elementu może być dokonana pod warunkiem, że w kolejce znajduje się minimum jeden element, czyli gdy kolejka nie jest pusta. Sprawdzenie, czy kolejka jest pusta polega na porównaniu obu wskaźników (odczytu oraz zapisu). Jeżeli oba wskaźniki pokazują to samo miejsce (mają identyczna zawartość) to kolejka jest pusta. W przeciwnym wypadku, kolejka zawiera elementy. Jednocześnie z tego wynika sposób wstępnego zainicjowania całej struktury. Przed zastosowaniem w sensie operacji zapisu lub odczytu elementów do kolejki należy ustawić wskaźniki, przykładowo nadać im wartość zero, chociaż w gruncie rzeczy nie jest istotne ile. Istotne jest, by było tyle samo (oczywiście w dopuszczalnym zakresie wynikającym z liczby elementów tablicy). To, że zwyczajowo realizuje się zainicjowanie wskaźników nadając początkowo im wartość zero, jest wyłącznie kwestią przyzwyczajeń.



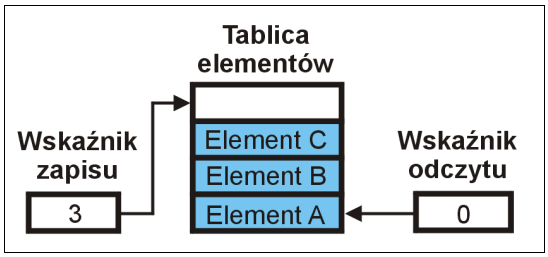
Ilustracja 7: Kolejka pusta



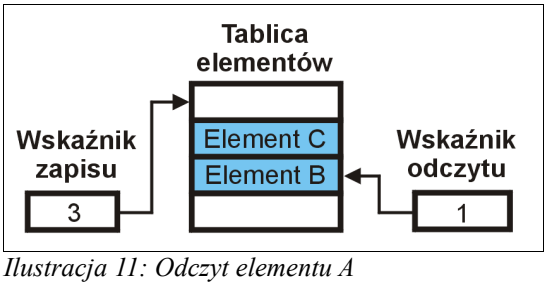
Ilustracja 8: Zapis elementu A



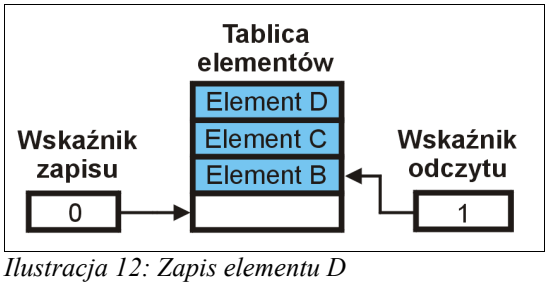
Ilustracja 9: Zapis elementu B



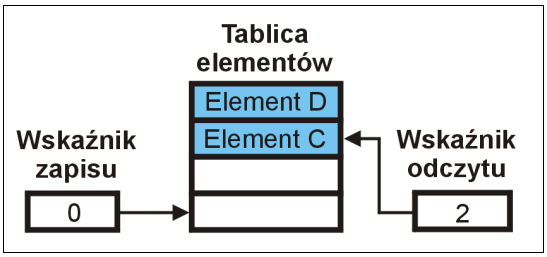
Ilustracja 10: Zapis elementu C



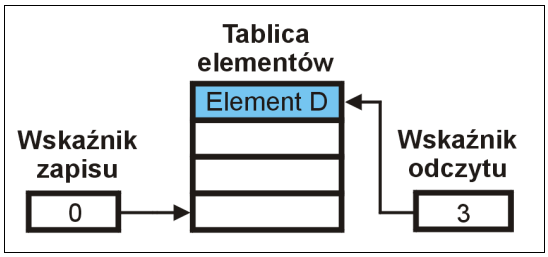
Ilustracja 11: Odczyt elementu A



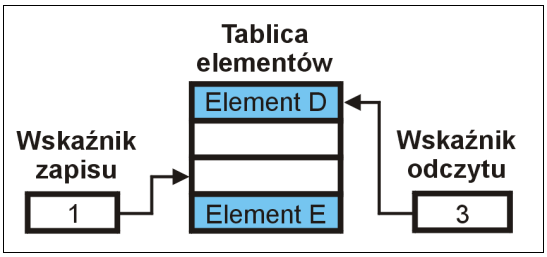
Ilustracja 12: Zapis elementu D



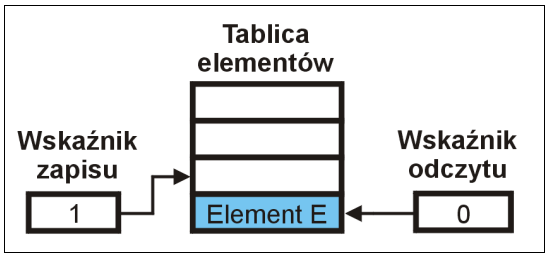
Ilustracja 13: Odczyt elementu B



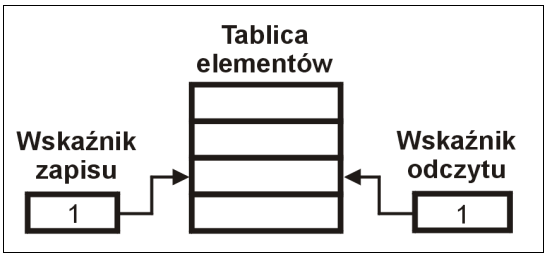
Ilustracja 14: Odczyt elementu C



Ilustracja 15: Zapis elementu E



Ilustracja 16: Odczyt elementu D



Ilustracja 17: Odczyt elementu E (kolejka pusta)

Ilustracje od 7 do 17 pokazują stan kolejki dla kilku operacji zapisu oraz odczytu informacji. Jak widać na ilustracjach, wskaźniki cyklicznie obiegają cała tablicę, stąd wywodzi się ich nazwa (bufory cykliczne). Powyższe struktury mają wiele zastosowań w oprogramowaniu mikrokontrolerów. Przykładowym zastosowaniem jest obsługa transmisji odbierania znaków z UART (kontrolera asynchronicznej transmisji szeregowej). Odebranie przez sprzęt znaku wywołuje przerwanie, w wyniku którego jest odczytany z rejestrów znak i umieszczony w buforze cyklicznym do późniejszego wykorzystania. Proponowana konstrukcja bufora cyklicznego może być następująca:

```
#define BuffSize                    32

typedef struct { uint8_t CycRdIndex ;
                 uint8_t CycWrIndex ;
                 uint8_t Buffer [ BuffSize ] ;
               } RXDRecT ;

static volatile RXDRecT RXDRec ;
```

gdzie w strukturze:

- *CycRdIndex* – jest wskaźnikiem odczytu,

- *CycWrIndex* – jest wskaźnikiem zapisu,
- *Buffer* – jest „ciałem" bufora jako tablica o określonej liczbie elementów.

oraz

zmienną o identyfikatorze *RXDRec*.

W takiej sytuacji operacja sprawdzenia, czy bufor cykliczny zawiera jakiekolwiek dane przedstawia się następująco:

```
uint8_t DataPresent ( void )
{
  if ( RXDRec . CycRdIndex == RXDRec . CycWrIndex )
    return ( FALSE ) ;
  else
    return ( TRUE ) ;
} /* DataPresent */
```

operacja zapisu elementu do kolejki:

```
void AddElement ( uint8_t RecVCh )
{
  RXDRec . Buffer [ RXDRec . CycWrIndex ] = RecVCh ;
  RXDRec . CycWrIndex ++ ;
  if ( RXDRec . CycWrIndex >= BuffSize )
    RXDRec . CycWrIndex = 0 ;
} /* AddElement */
```

operacja odczytu elementu z kolejki:

```
uint8_t GetElement ( void )
{
  uint8_t Ch ;
  /*-----------------------------*/
  Ch = RXDRec . Buffer [ RXDRec . CycRdIndex ] ;
  RXDRec . CycRdIndex ++ ;
  if ( RXDRec . CycRdIndex >= BuffSize )
    RXDRec . CycRdIndex = 0 ;
  return Ch ;
} /* GetElement */
```