Mikroprocesor jednoukładowy, czyli opowieść o tępym (aczkolwiek wyjątkowo pracowitym) urzędniku biurowym

część 2

Przed miesiącem, w pierwszej części artykułu, mikroprocesor (ATmega328) porównaliśmy do biura, w którym pracuje bardzo pracowity i skrupulatny, ale bardzo ograniczony umysłowo urzędnik. Opowieść zakończyliśmy na wielkim dzwonie (który umarłego by obudził) i który nie wiadomo czemu nosi nazwę RESET.

W biurze naszego urzędnika jest też kilka *mniejszych dzwonków*. Ich dźwięk również jest dość głośny, ale nie porażający. Te dzwonki informują, że ktoś z zewnątrz czegoś chce lub zgłasza jakiś problem, że ktoś lub coś chce obudzić z drzemki lub ***przerwać*** dotychczasową pracę urzędnika. Każde takie ***przerwanie***, oprócz dźwięku dzwonka, powoduje samoczynne podniesienie (podobno za pomocą jakiegoś sznurka, ale nie jest to dokładnie stwierdzone), czyli ***ustawienie*** odpowiedniej ***flagi*** (niektórzy mówią, że te patyczki flag są na zawiasach, bo można je *ustawiać* i kłaść – *zerować*). Mechanizm tych flag jest dodatkowo skomplikowany, bo można blokować i zezwalać na działanie każdej z osobna oraz wszystkich razem (*flag enable*).

Nasz urzędnik spokojnie, w uporządkowany sposób reaguje na „mały dzwonek”, czyli na ***zgłoszenie przerwania*** (***interrupt request***). Najogólniej biorąc, przerywa dotychczasową pracę, realizuje ***procedurę obsługi przerwania*** i wraca do wcześniejszej pracy. Dobrze rozróżnia, co to za przerwanie, a gdy jednocześnie zgłoszonych zostanie kilka, wtedy zaczyna od obsługi *przerwania o najwyższym priorytecie*, na podstawie tzw. ***priorytetu przerwań***. **Rysunek 5** przedstawia 19 przerwań o najwyższym priorytecie spośród 26 dostępnych.

Wbrew pozorom zasada obsługi przerwań jest prosta, jak wszystko, co potrafi zrealizować urzędnik: po *zgłoszeniu przerwania* ma on po prostu ***skoczyć*** do przegródki-komórki pamięci programu FLASH (Program Adress), wyznaczonej dla tego właśnie przerwania (co górnolotnie nazywa się ***wektorem przerwania***). Pod tym adresem znajdzie dalsze wskazówki, jak ma ***obsłużyć przerwanie***. W innych, większych biurach „adresy skoków przerwań” nie są ustalone na stałe, tylko jest dodatkowa kartka z adresami, gdzie trzeba skoczyć przy poszczególnych przerwaniach.

Stosownie do priorytetu najważniejszego ze zgłoszonych przerwań urzędnik ***skacze***, a właściwie sięga do odpowiedniej komórki pamięci FLASH. Ale nie robi tego gwałtownie i nagle. Najpierw spokojnie notuje na kartce aktualny numer zapisany w liczniku programu, żeby po obsłużeniu przerwania nie zgubić się i by wrócić do tego miejsca programu, gdzie był w chwili zgłoszenia przerwania. Po zapisaniu numeru licznika ***odkłada*** (popycha – ***push***) *na* ***stos*** (***stack***) kartkę z tym numerem-liczbą. Ewentualnie „zbiera też do kupy” dotychczasowe papiery pomocnicze i też może je odłożyć na stos. Rozkłada natomiast nowe papiery pomocnicze i *obsługuje przerwanie,* realizując stosowny kawałek programu. Gdy skończy obsługę, zeruje flagę danego przerwania i wraca do wcześniejszego zadania. W tym celu ***zdejmuje ze stosu*** (ściąga – ***pop***) papiery pomocnicze tego wcześniejszego zadania, wraca do zapisanego tam numeru licznika programu i realizuje wcześniejsze zadanie.

Rys. 5

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete

I dopiero tu widać, jak cenna jest drobiazgowa skrupulatność urzędnika. Niejeden dziwi się (Ty prawdopodobnie też), że też mu się chce tak bawić...

Ale w sumie to bardzo dobrze! Bo gdy następuje przerwanie, wtedy nasz kancelista skrupulatnie *odkłada na stos* adres zapisany na folii, ewentualnie także informacje z karteczek pomocniczych, określające stan biura. Dzięki temu po obsłużeniu przerwania może przywrócić wcześniejszy stan biura, a na folii (do licznika programu) znów zapisze numer komórki, do której trzeba skoczyć, żeby kontynuować wcześniejszy program.

Warto pamiętać, że dzwonki-przerwania przychodzą z zewnątrz, zwykle w przypadkowych, nieprzewidzianych momentach. Jedne mają wyższy priorytet, inne niższy. Jeżeli w trakcie obsługi przerwania zostanie zgłoszone nowe, *o wyższym priorytecie*, wtedy urzędnik zależnie od pewnych wcześniejszych ustaleń (ustawień) reaguje na różne sposoby.

Albo nie dokończy obsługi bieżącego przerwania, tylko znów zanotuje i *odłoży na stos* stan licznika programu i ewentualnie także opis aktualnego stanu biura, by obsłużyć przerwanie o wyższym priorytecie. Gdy obsłuży to pilne przerwanie, znów powróci do przerwania poprzedniego, o niższym priorytecie.

Albo też w momencie obsługi dowolnego przerwania wyłączy taką możliwość (*Global Interrupt Enable*) i wtedy zgłoszonymi nieco później przerwaniami (i ich ustawionymi flagami), niezależnie od ich priorytetu, zajmie się dopiero po obsłużeniu bieżącego.

W każdym razie urzędnik wie, że dzwonki-przerwania są bardzo ważne, więc przed powrotem do „normalnego

programu" obsługuje wszystkie zgłoszone przerwania, stosownie do ich priorytetu.

Warto dodać, że możliwa jest też praca dzwonków-przerwań bez flag, które zapamiętują ich wystąpienie. Jeżeli flagi sygnalizacyjne są wyłączone, wtedy nasz urzędnik reaguje na dzwonki tylko wtedy, gdy te dzwonią. Jeżeli „krótki dzwonek" zabrzmi, gdy jest on pilnie zajęty innym przerwaniem, to owo „krótkie przerwanie" nie zostanie obsłużone.

Dopiero w związku z problemem losowo zgłaszanych różnych przerwań widać, skąd nazwa *stos*. Jak widzisz, stos jest genialnie fajnym wynalazkiem, powalającym kolejno realizować kilka różnych zadań według ich ważności. Jednak trzeba pamiętać, że urzędnik jest ograniczony umysłowo i nie panuje nad całością. Niektórzy słusznie się boją, że gdy ten stos papierów będzie za wysoki, to się może przewrócić, a nisko pochylony nad papierami urzędnik (jest krótkowidzem) tego nie zauważy... I kłopot gotowy...

Urzędnik jest rzeczywiście krótkowidzem. Problem jest, ale nie do końca polega na przewróceniu się stosu, ponieważ (tu pewnie się zdziwisz) do realizacji stosu nasz urzędnik tak naprawdę wykorzystuje... kartki z komórek pamięci RAM, a ich jest stosunkowo niewiele, więc ich zasób może się wyczerpać. W każdym razie kwestii stosu nie można pozostawić przypadkowi. Pracę urzędnika trzeba tak zaplanować, żeby *nie przepełnić stosu*. W praktyce oznacza to głównie konieczność zarezerwowania na potrzeby stosu odpowiedniej liczby komórek pamięci RAM i korzystania z tak zwanego ***wskaźnika stosu*** (***stack pointer***).

Usprawnienia w pracy biura

Dawniej zarzucano naszemu urzędnikowi i jego kolegom, że często się mylą.

Nie wierz tym wierutnym kłamstwom! To są podłe pomówienia, rozsiewane przez tak licznych, nie tylko w naszym kraju, zwolenników teorii spiskowych!

Nasz urzędnik, choć tępy, jest wyjątkowo skrupulatny i w normalnych warunkach nie myli się nigdy. Jest tylko jeden wyjątek: może się pomylić, gdy jest wyjątkowo głodny i burczy mu w brzuchu. Ale to już nie jego wina, tylko ewidentne zaniedbanie firmy kateringowej obsługującej biuro. Ponieważ od dawna zdarzały się zaniedbania i problemy w tym zakresie, wprowadzono (podobno na żądanie jakichś rozsądnych związków zawodowych) sprzętowe zabezpieczenia i procedury, zwane ***BOD*** (***Brown-out Detector***), by urzędnik nie pracował „na głodniaka" i nie popełniał błędów.

Natomiast wspomniana wcześniej możliwość ***zawieszenia się*** wcale nie wynika z błędu urzędnika, tylko z błędu w programie (np. związanego z przepełnieniem stosu).

W każdym razie w normalnych warunkach pracy urzędnik nigdy się nie myli. Ponadto korzysta z szeregu bardzo cennych ułatwień i usprawnień, jakie wprowadzono w jego biurze.

I tak oprócz opisanej wcześniej folii (licznika programu), nasz pracowity urzędas ma też sporo innych karteczek pomocniczych, które ktoś kiedyś nazwał ***rejestrami*** (***registers***) i tak zostało. Ich zaletą jest to, że leżą na biurku tuż przed urzędnikiem i korzystanie z nich jest szybkie i wygodne. Dlatego na tych łatwo dostępnych karteczkach – *rejestrach* urzędnik notuje różne podręczne informacje bieżące, z których korzysta najczęściej. Z dostępnych w omawianym biurze 32 (ośmiobitowych) rejestrów ogólnego przeznaczenia korzysta on też podczas różnorodnych operacji i obliczeń. Niektóre łączy w pary, żeby z dwóch uzyskać jeden rejestr 16-bitowy. W szczególności często wykorzystuje 6 spośród nich do stworzenia „podwójnych" rejestrów zwanych X, Y, Z. Wykorzystuje je jako pamięć podręczną do przechowywania numerów – adresów w pamięci RAM oraz do tymczasowego przechowywania potrzebnych adresów rozkazów w pamięci FLASH przy tak zwanym *adresowaniu pośrednim*.

Na biurku stoi też kalkulator o specyficznej budowie. Kalkulator, na którego obudowie ktoś kiedyś grubym flamastrem napisał koślawe literki ***RISC–CPU*** może wykonywać tylko najprostsze działania arytmetyczne (dodawanie, odejmowanie w zakresie 0...65535, mnożenie liczb tylko w zakresie 0...255). Co ciekawe, ten kalkulator CPU w szczególnie bliski sposób połączony jest z rejestrami, w tym zwłaszcza z ***rejestrem stanu*** (***status register***), w którym oprócz ustawień dotyczących przerwań są też na bieżąco umieszczane „dodatkowe wyniki działań kalkulatora".

Wspomniane proste działania arytmetyczne ten dziwny kalkulator realizuje nie w znanym nam systemie dziesiętnym, tylko w systemie dwójkowym. Dziwny jest też dlatego, że potrafi realizować szereg specyficznych operacji, które nie są typowymi działaniami arytmetycznymi na liczbach, tylko operacjami na bitach (na poszczególnych zerach i jedynkach). Otóż jego ekscentryczne zamiłowanie do kodu dwójkowego skutkuje tym, że biegle potrafi przeprowadzać różne tak zwane *operacje bitowe*. Ośmiobitowe „kawałki" z pamięci RAM chętnie traktuje nie jako liczby dwójkowe, tylko jako ciągi niezależnych bitów. I albo jednocześnie na takich zestawach ośmiu niezależnych bitów, albo na pojedynczych bitach przeprowadza specyficzne operacje, które nam z początku wydają się dziwne, a które odgrywają dużą rolę w codziennej pracy urzędnika.

Sterowanie przebiegiem programu

Jak już wiesz, urzędnik normalnie ma pobierać rozkazy z kolejnych przegródek-komórek pamięci FLASH. Wiesz, że taką pracę mogą zaburzyć „czynniki zewnętrzne": przerwania lub wielki dzwon – RESET.

Ale najogólniej biorąc, realizacja programu polega na tym, że urzędnik odczytuje rozkazy z kolejnych komórek-przegródek pamięci FLASH. Rano pracę zaczyna od pierwszej komórki (o numerze-adresie zero), ale zwykle już w pierwszej) znajduje ***rozkaz skoku*** do komórki pamięci FLASH o innym numerze-adresie, czyli w sumie jest to skok do rozkazu, znajdującego się gdzieś w pamięci programu. Te *rozkazy skoku* mogą być odmienne, a adres-numer komórki, do której trzeba ostatecznie skoczyć, może być podany mniej lub bardziej bezpośrednio. Na przykład w pamięci FLASH może być kod: rozkaz skoku oraz w tym samym rozkazie – numer-adres komórki pamięci FLASH, pod który trzeba skoczyć. Wtedy mamy tzw. ***skok bezwarunkowy*** (***unconditional jump***) i ***adresowanie bezpośrednie***.

Ale często jest inaczej i wiele osób ma problem ze zrozumieniem, czym jest ***adresowanie pośrednie*** i związane z tym ***wskaźniki***. A w sumie jest to proste (jak wszystko, co robi nasz urzędniczyna), bowiem podstawowa zasada adresowania pośredniego jest taka: w rozkazie w pamięci FLASH też jest zawarty adres, ale nie ten adres w pamięci FLASH, dokąd docelowo trzeba skoczyć. Otóż w rozkazie zawarty jest numer-adres pamięci RAM, a dopiero w pamięci RAM o tym numerze znajduje się właściwy adres, do którego trzeba ostatecznie skoczyć w pamięci FLASH (albo liczba komórek pamięci FLASH, którą trzeba „przeskoczyć"). Trochę dziwne, prawda?

Dziwne, ale logiczne, bardzo często wykorzystywane w praktyce i wcale nie tak trudne, jeżeli potrafi to zrealizować nasz przygłupiasty urzędnik...

Potrafi, ale tylko dlatego, że ma w programie odpowiednie rozkazy. Trochę trudniejsze jest to dla programisty, który „ustawia pracę urzędnika". Na szczęście programista zamiast pisać numery-adresy komórek pamięci, genialnie ułatwia sobie pracę, wykorzystując ***zmienne*** (***variables***) oraz ***stałe***, ale to odrębny ogromnie ważny wątek, który omówimy oddzielnie.

Wracamy do sterowania programem. Co bardzo ważne, nasz urzędnik potrafi ***sprawdzać*** proste ***warunki***, a w szczególności sprawdzać, czy liczba jest równa zeru, czy nie oraz ***porównywać*** liczby (większa, mniejsza, równa). Tak naprawdę, robi to za niego kalkulator i te „dodatkowe wyniki porównania" automatycznie dostępne są w *rejestrze stanu*.

Przy okazji wspomnijmy, że urzędnik nie ma wprawdzie poczucia sprawiedliwości, ale rozróżnia takie ważne pojęcia jak PRAWDA i FAŁSZ, tylko je dziwnie upraszcza i wiąże z liczbami (wartość zero traktuje jako FAŁSZ, a każdą inną wartość – jako PRAWDĘ). Potrafi porównywać, a także sprawdzać swoją wersję PRAWADA/FAŁSZ, a potem stosownie do tego reagować.

To sprawdzanie warunków oraz odpowiednie reakcje są bardzo ważne w praktyce, ponieważ pozwalają sterować pracą programu. Mianowicie niektóre z realizowanych przez urzędnika rozkazów to ***skoki warunkowe*** (***conditional jump***). Najprościej mówiąc, są wykonywane tylko wtedy, gdy spełniony jest dany warunek. Różne *skoki warunkowe* pozwalają „rozgałęzić" działanie programu, zależnie od różnych czynników czy wyników wcześniejszych operacji.

Na marginesie można też wspomnieć, że oprócz nieco prostszych „zwykłych" skoków (jumps), urzędnik może wykonać „złożony skok", zwany ***wywołaniem podprogramu*** (***subroutine call***), który oprócz skoku dodatkowo wiąże się z użyciem stosu, więc jest jakby „wewnętrznym, programowym przerwaniem".

Podsumujmy: podczas pracy zasadniczo realizowane są rozkazy zawarte w kolejnych komórkach pamięci programu FLASH, ale można to zmienić zarówno za pomocą zdarzeń zewnętrznych (reset i przerwania), jak też wykorzystując zawarte w programie rozkazy skoków bezwarunkowych i warunkowych.

Sposób pracy

I tu dochodzimy do specyfiki pracy naszego urzędnika. Jest on nieprawdopodobnie pracowity i nieprawdopodobnie szybki. Potrafi w ciągu sekundy zrealizować nawet do kilkunastu milionów (tak!) elementarnych rozkazów. Dawniej, w czasach przed kryzysem, nie zawracano sobie głowy kwestiami oszczędności energii. Dopiero od niedawna coraz częściej wykorzystuje się dostępne w biurze możliwości snu (*Powerdown*) i różnych odmian drzemki (*Idle*). Jednak nadal najczęściej sposób pracy urzędnika można podsumować tak: „*nawet jeśli nic nie robi, to też coś robi*": cały czas odczytuje z pamięci i realizuje kolejne rozkazy. Aby zająć pracowitego urzędnika, podstawą praktycznie każdego programu jest tak zwana ***pętla*** (po angielsku ***loop***). Zasadniczo nasz urzędnik wykonuje rozkazy z kolejnych przegródek-komórek pamięci FLASH. Jednak główny program jego działania zawiera w komórce o jakimś numerze-adresie N rozkaz, którego sens w najprostszym przypadku jest taki: *skocz (wróć) do komórki o adresie N*. Nasz przygłupiasty urzędnik nie zbuntuje się przeciwko takiemu debilnemu zadaniu i z zapałem godnym lepszej sprawy gotów jest w takiej *nieskończonej pętli* w kolejnych taktach zegara mozolnie odczytywać zawartość komórki N i wykonywać zawarty tam rozkaz skoku. Taki program tylko na pozór nie ma żadnego sensu, jednak bywa bardzo często wykorzystywany, bowiem w ten sposób czeka na ewentualne przerwania zewnętrzne.

Podobnie w *nieskończonej pętli* działa program, który w uproszczeniu można streścić tak: *sprawdź, czy jest coś do zrobienia i jeśli nie ma nic do zrobienia, to skocz (wróć) do komórki o adresie N*.

Do tych ważnych zagadnień będziemy wracać. A na razie omówimy kolejne ogromnie ważne zagadnienie....

Obwody peryferyjne

Większość kolegów naszego urzędnika pracuje z biurach (klasycznych procesorach), które praktycznie nie mają kontaktu ze światem zewnętrznym. Nasz urzędnik ma szczęście lub nieszczęście pracować w specyficznym biurze (*mikrokontroler jednoukładowy*), które ma szerokie możliwości kontaktów ze światem zewnętrznym. Budynek biura ma wprawdzie strasznie grube ściany, ale podobno są w nich swego rodzaju „kanały" i inne bardziej skomplikowane urządzenia do łączności ze światem. Choć biuro wcale nie jest położone nad wodą, „kanały" te nie wiadomo dlaczego nazywane są ***portami***.

A te porty są bardzo dziwne i mało kto dokładnie poznał wszystkie szczegóły ich budowy. Niektórzy wyobrażają je sobie jako zestawy otworów o centymetrowej średnicy wywiercone w grubych ścianach. ***Port*** to zespół (ośmiu) otworów, a pojedynczy „otwór na świat" jak na złość nosi określenie – ***pin***. Urzędnik mógłby je potraktować jako ***wyjścia***, na przykład przez otwór mógłby wystawić na zewnątrz czarną albo białą kulkę i w ten sposób przekazać na zewnątrz informację (0/1).

Ale są to otwory – ***porty dwukierunkowe*** i urzędnik mógłby przez nie „obserwować świat". Jednak otwory mają bardzo małą średnicę, ściany są grube, więc nawet gdyby ktoś przyłożył oko do otworu, to nie widać szerszego fragmentu świata, tylko bardzo wąski wycinek. Tak naprawdę, to nie widać świata, tylko można rozróżnić dwa stany: jasno/ciemno i zapisać na kartce jako 0 lub 1.

Z jakichś nie dla wszystkich z powodów, między innymi historycznych, przy pracy pinu w roli wejścia można podciągnąć do góry na gumce zewnętrzną zasłonkę – kawałek czarnego plastiku. Wtedy normalnie otwór od zewnątrz jest zasłonięty czarnym tworzywem i urzędnik, gdyby chciał sprawdzić stan wejścia, odnotuje ciemność, czyli stan 1. Wszyscy przyzwyczaili się mówić, że wtedy ***wejście jest podciągnięte*** (***pull-up***). Tylko wtedy, gdy ktoś lub coś z zewnątrz ściągnie ten plastik w dół, w kierunku ziemi, można zobaczyć światło, czyli stan 0.

A biurach omawianego typu, zależnie od ich wielkości, może być od kilku do kilkudziesięciu takich pinów pogrupowanych w ośmiobitowe ***porty wejścia/wyjścia*** (***I/O-Ports***).

Teoretycznie urzędnik mógłby je obsługiwać w opisany sposób, ale tego nie robi. Przedstawiony opis „otworów" to dramatycznie uproszczone wyobrażenia o tym, jak są zbudowane te porty. Otóż niektóre piny dodatkowo zawierają dużo bardziej skomplikowane mechanizmy. Niektóre porty mają jakiś dodatkowy związek z zegarem, inne z przerwaniami, jeszcze inne z jakimiś dziwnymi sposobami komunikacji, czyli z przesyłaniem podczas pracy informacji z biura na zewnątrz i odwrotnie.

Bardzo prawdopodobne są obiegowe poglądy, że szczegóły budowy mogliby zrozumieć tylko elektronicy (czytający EdW), ale na pewno nie biedny, ograniczony umysłowo urzędnik. I chyba tylko dlatego, żeby go nie przytłoczyć i nie załamać, projektanci biura wykorzystali pewien szatańsko sprytny pomysł.

Zdradzę Ci teraz ważną tajemnicę: urzędnik nie tylko nie rozumie działania, ale w ogóle nie wie o istnieniu portów i innych jeszcze bardziej skomplikowanych urządzeń wyposażenia biura. On by się na pewno załamał, gdyby wiedział, jak to wszystko jest skomplikowane. Dlatego wszystkie te dodatkowe urządzenia peryferyjne (*peripherials*) są wbudowane w ściany, nie widać ich od środka, więc urzędnik nawet nie domyśla się ich istnienia. Nie ma na przykład pojęcia o budowie tak zwanego przetwornika analogowo-cyfrowego (oznaczanego A/C lub ADC), czy o budowie kilku liczników – timerów ani o interfejsach UART, SPI, TWI...

Owszem obsługuje je, ale bez zrozumienia – bezrefleksyjnie realizując elementarne rozkazy programu.

Jak to możliwe?

Otóż już wcześniej mówiliśmy o 32 ośmiobitowych *rejestrach ogólnego przeznaczenia* (*general purpose registers*), będących dla urzędnika karteczkami, które są „blisko pod ręką” i które wykorzystuje najczęściej.

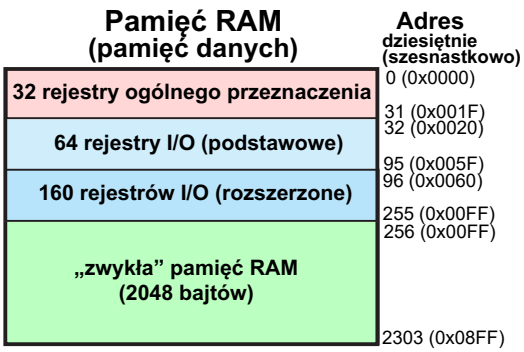
Oprócz tego (zależnie od wielkości i wyposażenia biura) istnieje kilkadziesiąt innych karteczek-rejestrów, ściśle związanych właśnie z portami i innymi wymienionymi urządzeniami. Zasadniczo rejestry mogą być (i są) szczególnymi, złożonymi konstrukcjami, jednak w punktu widzenia urzędnika rejestry można traktować dokładnie tak samo, jak karteczki/komórki pamięci RAM

Nie tylko można, ale właśnie tak to jest zorganizowane w tym biurze! Wspomniany szatańsko sprytny, a prosty w swej istocie pomysł polega na tym, że *rejestry są komórkami pamięci RAM*, przynajmniej z punktu widzenia urzędnika.

Zdyscyplinowany urzędnik najprawdopodobniej nigdy tego nie zrobi, ale gdyby kiedyś dokładnie obejrzał przegródki pamięci RAM, to zauważyłby, że niektóre są bardzo dziwne. Zorientowałby się, że te o numerach 0...31 są ściśle powiązane z kalkulatorem (CPU). Jeszcze dziwniejsze są przegródki RAM o numerach 32 do 255. I tak do niektórych nie można wpisywać, a tylko odczytywać z nich dane, które się tam pojawiają bez udziału urzędnika. W niektórych nie ma pełnowartościowych ośmiu bitów (niektórych bitów nie można ani zapisywać, ani odczytywać). Dotyczy to też wielu całych komórek-bajtów RAM – miejsce jest zarezerwowane, jakby ktoś je przewidział do przyszłej rozbudowy. **Rysunek 6** pokazuje podział przestrzeni adresowej pamięci RAM w biurze ATmega328.

Urządzenia peryferyjne mogą być (i są) bardzo skomplikowane, a urzędnik nie ma o tym bladego pojęcia, ponieważ do ich sterowania i odczytywania z nich danych służą wyznaczone komórki pamięci RAM. Obsługuje je więc dokładnie tak, jak inne komórki pamięci RAM, nie mając bladego pojęcia o ich budowie i szczegółach działania.

Rys. 6



Urzędnik trwa więc w błogiej nieświadomości tego, co tak naprawdę robi. Tymczasem wykorzystując tzw. operacje bitowe, można dowolnie modyfikować i odczytywać stan albo wszystkich ośmiu, albo poszczególnych pojedynczych pinów portu. Co ważne, w danym czasie niektóre piny danego portu mogą być wejściami a inne w tym samym czasie wyjściami, w dowolnej konfiguracji. Do obsługi danej ósemki pinów (portu) nie wystarczy jeden rejestr. Potrzebne są trzy rejestry: jeden do określania kierunku pracy (wejście albo wyjście), drugi do ustawiania stanu pinów, trzeci do odczytywania stanu pinów.

I nasz urzędnik nie wie, że wpisując zera i jedynki do komórki RAM o numerze-adresie 36 (0x24) konfiguruje osiem bitów portu oznaczonego B, by wszystkie lub poszczególne bity pracowały jako wyjścia lub wejścia. Nie wie, że wpisując bajt danych do komórki o adresie 37 (0x25) określa stan logiczny pinów, gdy pracują jako wyjścia oraz włączając wspomniane podciąganie, gdy pracują jako wejścia. I wreszcie nie wie, ze odczytując zawartość komórki pamięci RAM o adresie 35 (0x23) w rzeczywistości odczytuje aktualny stan pinów portu B.

On nie wie, że w jego biurze (ATmega348) są trzy takie porty. Nie ma też bladego pojęcia, że na przykład sześć pinów portu C może pełnić zupełnie inną rolę, mianowicie sześciu wejść przetwornika analogowo-cyfrowego. Owszem, jeżeli w programie znajdzie odpowiednie rozkazy, to skonfiguruje przetwornik ADC, wpisując odpowiednie dane do komórek RAM o numerach 122...124 (0x7A...0x7C), a potem z komórek o numerach 120, 121 (0x78, 0x79) odczyta 10-bitowy wynik pomiaru.

Podobnie zupełnie nieświadomie skonfiguruje i obsłuży timery/liczniki i interfejsy komunikacyjne, zgodnie z programem wpisując i odczytując dane do odpowiednich komórek pamięci RAM. W ten sposób bezmyślnie wykonując program, składający się z elementarnych rozkazów, zrealizuje skomplikowane zadania zupełnie nie wiedząc, co tak naprawdę robi.

Na koniec jeszcze jedna sprawa. Biuro naszego urzędnika należy do większych i bardziej nowoczesnych. A jemu samemu udostępniono dodatkową możliwość. Mianowicie w innych podobnych biurach gotowe zafoliowane kartki z rozkazami do pamięci FLASH i EEPROM trzeba podać z zewnątrz i poukładać w przegródkach bez udziału urzędnika. Zazwyczaj wiąże się z tym poważna przerwa w pracy biura i pewne kłopoty logistyczne. Jednak z gruntu nieprawdziwe są lamerskie plotki, że karteczki te wciskane są w szczelinę pod drzwiami – w istocie wykorzystywane są do tego wspomniane porty i to albo w (popularniejszym i prostszym) trybie szeregowym, albo w trybie równoległym, w każdym razie bez udziału naszego urzędnika.

Natomiast w omawianym biurze ATmega328 przewidziano możliwość, by urzędnik sam „wciągnął” czy „zassał” z zewnątrz program oraz sam przygotował i poukładał zafoliowane karteczki w pamięciach FLASH i EEPROM. Oczywiście jest zbyt ograniczony, by sam z siebie podjął takie działania, więc przewidziano możliwość zarezerwowania „najwyższych” adresów pamięci FLASH, by tam trwale umieścić nieduży program, nazywany ***bootloader***, którego realizacja pozwoli urzędnikowi gładko „wciągnąć” program roboczy z zewnątrz, bez znaczącego zakłócania działalności biura. W razie potrzeby urzędnik przerwie więc normalną pracę, „wciągnie” do pamięci FLASH zupełnie nowy program, cały czas będąc nieświadomy, że z pamięci FLASH wydzielona jest niewielka część na *bootloader*, i że do każdej są oddzielne „fusy” konfigurujące.

Praca naszego urzędnika nie jest więc frustrująca, bowiem on ma tylko sumiennie i szybko wykonywać elementarne rozkazy, o których sensie nie ma bladego pojęcia. Zostawimy go w błogiej nieświadomości. Niech robi swoje, a mnóstwa szczegółów związanych ze swą pracą on nigdy nie zrozumie.

Nie przejmuj się, jeśli i Ty wszystkiego nie rozumiesz! I tak jesteś w nieporównanie lepszej sytuacji, niż słabo rozgarnięty urzędnik. On prawie nic nie rozumie, a skutecznie działa. A Ty, jeśli tylko chcesz, możesz z czasem zrozumieć wszystkie szczegóły pracy mikroprocesorów. Także i Ty, zaczynając programować mikroprocesory, nie musisz wiedzieć wszystkiego. Na początek będziesz wykorzystywał domyślne ustawienia fabryczne i zamiast pisać programy od zera, możesz modyfikować „gotowce”. Dlatego nie stresuj się stopniem skomplikowania choćby obwodów zegarowych, watchdoga, timerów czy systemu przerwań.

Kończymy opowieść o urzędniku biurowym i jego biurze, a dalsze rozważania o procesorach AVR będziemy realizować w konwencji poważniejszej.

Mam nadzieję, że przedstawiony materiał zmniejszy obawy przed programowaniem mikroprocesorów. A z tej opowieści zapamiętaj przede wszystkim sprytny pomysł, by rejestry do obsługi urządzeń peryferyjnych traktować i obsługiwać jak „zwykłe” komórki pamięci RAM.

Piotr Górecki