STUDIA INFORMATICA 2013
Volume 34 Number 3 (113)

Roman MIELCAREK
Politechnika Poznańska, Instytut Informatyki

PROGRAMOWE MOŻLIWOŚCI NADMIAROWEGO KODOWANIA DANYCH W KOMPAKTOWYCH STEROWNIKACH PLC FIRMY MITSUBISHI ELECTRIC

Streszczenie. W artykule przedstawiono różne metody programowego kodowania nadmiarowego danych, mogące mieć zastosowanie w protokołach użytkownika, implementowanych w sterownikach PLC typu FX firmy Mitsubishi Electric. Rozpatrzono przykłady kodów: definiowanych układem równań, iterowanych i w zapisie wielomianowym. Dla każdej z rozpatrywanych metod kodowania przedstawiono fragmenty przykładowych programów, realizujących te metody w sterowniku FX.

Słowa kluczowe: kod nadmiarowy, protokół transmisyjny, sterownik PLC

SOFTWARE CAPABILITIES OF REDUNDANT DATA ENCODING IN MITSUBISHI ELECTRIC PLC COMPACT CONTROLLERS

Summary. This paper presents several methods of redundant data encoding, which can be applied in user protocols implemented in Mitsubishi Electric FX programmable controllers. We consider the code examples: defined with systems of equations, iterated, and in polynomial notation. For each of the encoding methods we provide program snippets showing the implementation of these methods in an controller.

Keywords: redundant code, transmission protocol, PLC

1. Wprowadzenie

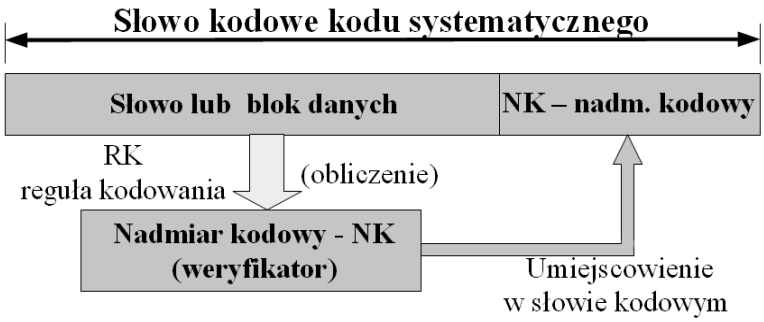
Sterowniki PLC typu FX firmy Mitsubishi Electric [1], [2] mają otwarty programowo dostęp do konfiguracji własnych lub dobudowanych portów szeregowych. Porty te można skonfigurować w jeden z trzech standardowych styków transmisyjnych typu RS. Ta dostępność daje szerokie możliwości zaimplementowania własnego protokołu transmisyjnego (protokołu

użytkownika) [4], jeśli protokoły wbudowane nie są satysfakcjonujące dla danej aplikacji. Zastosowanie protokołu użytkownika sprawia, że jest konieczne wprowadzenie nadmiarowego kodowania transferowanych danych w nadajniku i ich weryfikacja (dekodowanie) po stronie odbiornika. Celem tego procesu jest uzyskanie możliwości wykrywania błędów w odbieranych danych, aby zapobiec ich przekazaniu do obiektu ich przeznaczenia.

Sterowniki FX mają kilka instrukcji zaawansowanych [2] ogólnego przeznaczenia, które można wykorzystać w procesie programowego kodowania, a także dekodowania danych. Oprócz nich mają również kilka instrukcji specjalizowanych, przydatnych w wybranej metodzie kodowania. Każdą z nich przedstawiono przy rozpatrywaniu danej metody kodowania.

2. Systematyczne kody liniowe

Kodem nadmiarowym nazywana będzie dalej taka struktura danych, w której do k bitów danych dodane zostanie w procesie kodowania r bitów nadmiarowych. Postać (określenie kombinacji) tych r bitów nadmiarowych definiuje reguła kodowania. Przyjęte zostanie również dalej, że te r bitów nadmiarowych zależeć będzie tylko od aktualnego bloku tych k bitów danych, a nie od danych, które zostały wcześniej nadane. Tak więc blokowy kod nadmiarowy można scharakteryzować tymi dwoma parametrami i zapisać w postaci $(k+r, k) = (n, k)$, gdzie liczba n określa sumaryczną liczbę bitów danych i bitów nadmiarowych. Ten blok n bitów przedstawiany będzie w postaci n-bitowego słowa, zwanego słowem kodowym. Jeśli przyjąć, że bity danych (zwane również pozycjami informacyjnymi) występują na początku tego słowa kodowego (numerując od jego lewej strony), a pozycje nadmiarowe (wprowadzone w procesie kodowania) są końcowymi pozycjami tego słowa, to tak określony kod nazywany będzie dalej kodem systematycznym. Sposób określania pozycji nadmiarowych w kodzie systematycznym przedstawiono na rys. 1.



Rys. 1. Schemat układu regulacji
Fig. 1. Scheme of control system

Wśród możliwych reguł kodowania określających pozycje nadmiarowe, największy zbiór stanowią kody liniowe, definiowane układem równań z operacją liniową w postaci sumy mod 2 elementów bitowych słowa kodowego. Układ tych równań przedstawiono w skrótowym zapisie we wzorze (1).

$$\sum_{j=1}^{n} Sj \cdot Hjl = 0 \quad \text{dla} \quad l = 1, 2, \ldots, r, \tag{1}$$

gdzie:

$\sum$ – operator wieloargumentowej sumy mod 2,

Sj – j-ta pozycja słowa kodowego,

Hjl – współczynnik {0, 1} określający przynależność pozycji Sj do l-tego równania.

W powyższym układzie równań numeracja pozycji danych odbywa się dla $1 \leq j \leq k$, gdzie pozycja j=1 wskazuje zazwyczaj najstarszą (w zapisie systemowym) pozycję słowa danych. Pozostałe numery od k+1 do n określają pozycje nadmiarowe. W kodzie systematycznym każde równanie powyższego układu zawiera tylko jedną pozycję nadmiarową. Współczynniki Hjl zapisane w postaci tablicy, tak jak występują w kolejnych równaniach, tworzą tzw. macierz kontrolną kodu H. Swą nazwę bierze ona od pozycji kontrolnych – nadmiarowych, które definiuje. Przykład macierzy kontrolnej dla rozpatrywanego dalej kodu o parametrach (n, k) = (12, 8) przedstawiono na rys. 2.

Zbiór słów kodowych kodu liniowego, określony według równań (1), tworzy strukturę algebraiczną, zwaną grupą. Struktura ta spełnia pięć aksjomatów [4], z których dwa najważniejsze dla teorii kodowania liniowego są następujące:

- *A1* – grupa zawiera słowo zerowe *Sz* (słowo złożone z samych zer – słowo neutralne),
- *A2* – suma mod 2 dwóch dowolnych słów kodowych (suma po odpowiadających sobie pozycjach) jest również słowem kodowym ze zbioru grupy.

Aksjomat *A1* określa warunek konieczny przynależności danego zbioru słów kodowych do klasy kodów liniowych, natomiast aksjomat *A2* (aksjomat zamkniętości) pozwala określić zbiór słów kodowych kodu liniowego na bazie podzbioru słów, cechujących się tzw. niezależnością liniową. Podzbiór ten ma tę własność, że dowolnego słowa z tego podzbioru nie da się wygenerować z pozostałych opierając się na aksjomacie *A2*. Podzbiór ten, zapisany w postaci tablicy, nosi nazwę macierzy generującej G. Macierze H oraz G, dla przykładowego kodu o parametrach (n, k) = (12, 8), przedstawiono na rys. 2. Parametr $r = n - k$ określa liczbę pozycji kontrolnych w słowie kodowym.

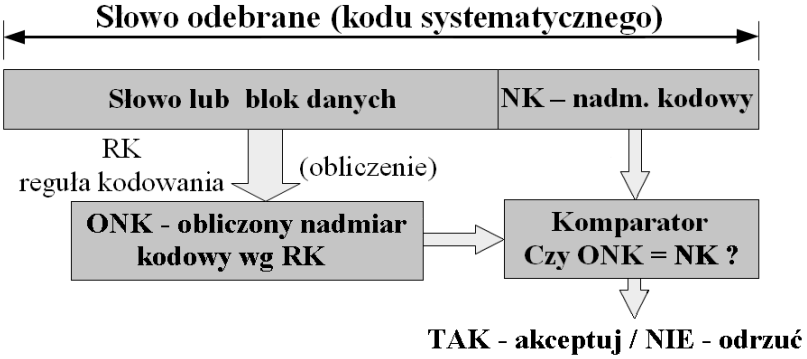
$$\boldsymbol{H} = \begin{bmatrix} 1010\ 1010\ 1000 \\ 0101\ 0101\ 0100 \\ 1100\ 1100\ 0010 \\ 0011\ 0011\ 0001 \end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix} 1000\ 0000\ 1010 \\ 0100\ 0000\ 0110 \\ 0010\ 0000\ 1001 \\ 0001\ 0000\ 0101 \\ 0000\ 1000\ 1010 \\ 0000\ 0100\ 0110 \\ 0000\ 0010\ 1001 \\ 0000\ 0001\ 0101 \end{bmatrix}$$

$$\boldsymbol{H}_{r\times n} = [\boldsymbol{P}_{r\times k}, \boldsymbol{I}_{r\times r}] \qquad \boldsymbol{G}_{k\times n} = [\boldsymbol{I}_{k\times k}, \boldsymbol{P}^{T}_{k\times r}]$$

Rys. 2. Macierz kontrolna H i macierz generująca G przykładowego kodu $(n, k) = (12, 8)$
Fig. 2. The control matrix H and the generator matrix G of a sample code $(n, k) = (12, 8)$

Z powyższego przykładu wynika, że kod liniowy może być definiowany dwoma różnymi metodami: za pomocą macierzy *H* (układu równań) lub za pomocą macierzy *G* (podzbiorem liniowo niezależnych słów kodowych). Obie macierze (oba sposoby definiowania kodu) łączy podmacierz *P*, występująca w postaci prostej w macierzy *H* i transponowanej w macierzy *G*.

Wprowadzone pozycje nadmiarowe (kontrolne) w nadajniku służą do weryfikacji w odbiorniku odebranego słowa kodowego na podstawie zastosowanej reguły kodowania. Proces kontroli (dekodowania) przedstawiono na rys. 3.



Rys. 3. Schemat procesu dekodowania słowa kodowego w odbiorniku
Fig. 3. The block diagram of the decoding process of the code word in the receiver

Podjęcie decyzji NIE w dekoderze odbiornika ma miejsce w przypadku wykrycia błędu w odebranym słowie kodowym, czyli wystąpienie tzw. błędu wykrytego w procesie transmisji. Decyzja TAK oznacza zgodność obliczonego nadmiaru kodowego ze stosowaną regułą kodowania. Ta zgodność występuje w dwóch przypadkach:

- gdy odebrane słowo kodowe jest bezbłędne – identyczne jak zostało nadane w nadajniku,
- gdy wystąpił tzw. błąd niewykryty – nadane słowo kodowe zostało przekłamane w inne słowo kodowe, spełniające również regułę kodowania.

Problem tworzenia reguły kodowania sprowadza się przede wszystkim do uzyskania maksymalnej liczebności zbioru błędów wykrytych dla danych parametrów kodu (n, k).

3. Nadmiarowe kodowanie danych w sterowniku PLC

Decyzja o zastosowaniu danego kodowania danych w danym systemie transmisji wymaga rozpatrzenia wielu aspektów tego systemu i trudno tu podać jakąś uniwersalną receptę (algorytm), pozwalającą uzyskać zamierzony cel. Na pewno na tym etapie decyzyjnym należy rozpatrzyć:

- jakie dane będą podlegały kodowaniu – czy pojedyncze słowa systemowe czy też ich większa liczba (bloki danych),
- na jakie błędy kod ma być odporny – na błędy niezależne o określonej krotności (minimalna odległość Hamminga) czy też na błędy seryjne (paczki błędów) o określonej długości.

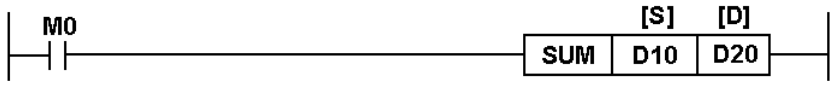
Najtrudniejsza jest odpowiedź na drugie pytanie, gdyż wymaga ona badania statystycznego zakłóceń przyszłego kanału transmisyjnego, który najczęściej jest nieznany. Dlatego też dalej rozpatrywany będzie tylko pierwszy wymieniony wyżej problem, czyli kodowanie i dekodowanie pojedynczych słów danych lub ich bloków.

3.1. Kodowanie i dekodowanie pojedynczych słów danych

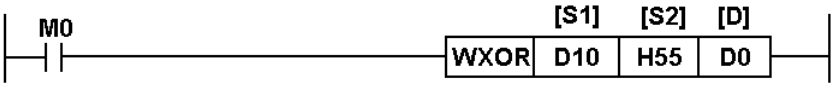
W przykładzie kodu nadmiarowego, przedstawionym za pomocą macierzy H i G na rys. 2, liczba pozycji nadmiarowych wynosi 4. Zastosowanie takiego kodu w systemie transmisji o systemowym słowie 8-bitowym, jakim jest słowo danych byłoby nieefektywne z tego powodu, że podczas transmisji słowa nadmiarowego 4 pozycje tego słowa byłyby niewykorzystane. Stąd wynika prosta implikacja, że liczba bitów nadmiarowych powinna być zawsze pełną wielokrotnością długości słowa danych. Dla wartości $r = k$ prowadzi to do kodu o parametrach $(n, k) = (2k, k)$, dla którego obie macierze H i G są tych samych rozmiarów, co jest przypadkiem szczególnym. Dlatego też dalej będzie rozpatrywany przykład kodowania i dekodowania dla $r = 4$, gdyż kod z $r = 8$ miałby tylko o cztery równania więcej.

Obliczenie danej pozycji nadmiarowej, wg równań z rys. 2, wymaga obliczenia sumy mod 2 z wybranych pozycji słowa danych. Sterowniki FX pozwalają na takie obliczenie metodą pośrednią z wykorzystaniem instrukcji SUM (*Sum of active bits*), przedstawionej w języku drabinkowym na rys. 4a. Instrukcja ta, jeśli warunek wykonania M0 = „1”, dokonuje sumowania wszystkich 16 pozycji rejestru źródłowego [S] = D10 o wartości „1” i umieszcza wynik sumowania w rejestrze celu [D] = D20. Najmłodszy bit rejestru celu jest zatem bitem parzystości (0) lub nieparzystości (1) liczby pozycji „1” w słowie źródłowym. Jest więc poszukiwaną wartością danej pozycji nadmiarowej.

a) **SUM: obliczenie liczby pozycji o stanie "1" z rejestru źródłowego [S] w rejestrze celu [D]**



b) **WXOR: obliczenie sumy EXOR słów rejestrów źródłowych [S1] i [S2] w rejestrze celu [D]**



Rys. 4. Instrukcje sterownika FX przydatne przy kodowaniu i dekodowaniu danych
Fig. 4. The instructions of an FX controller useful in data coding and decoding

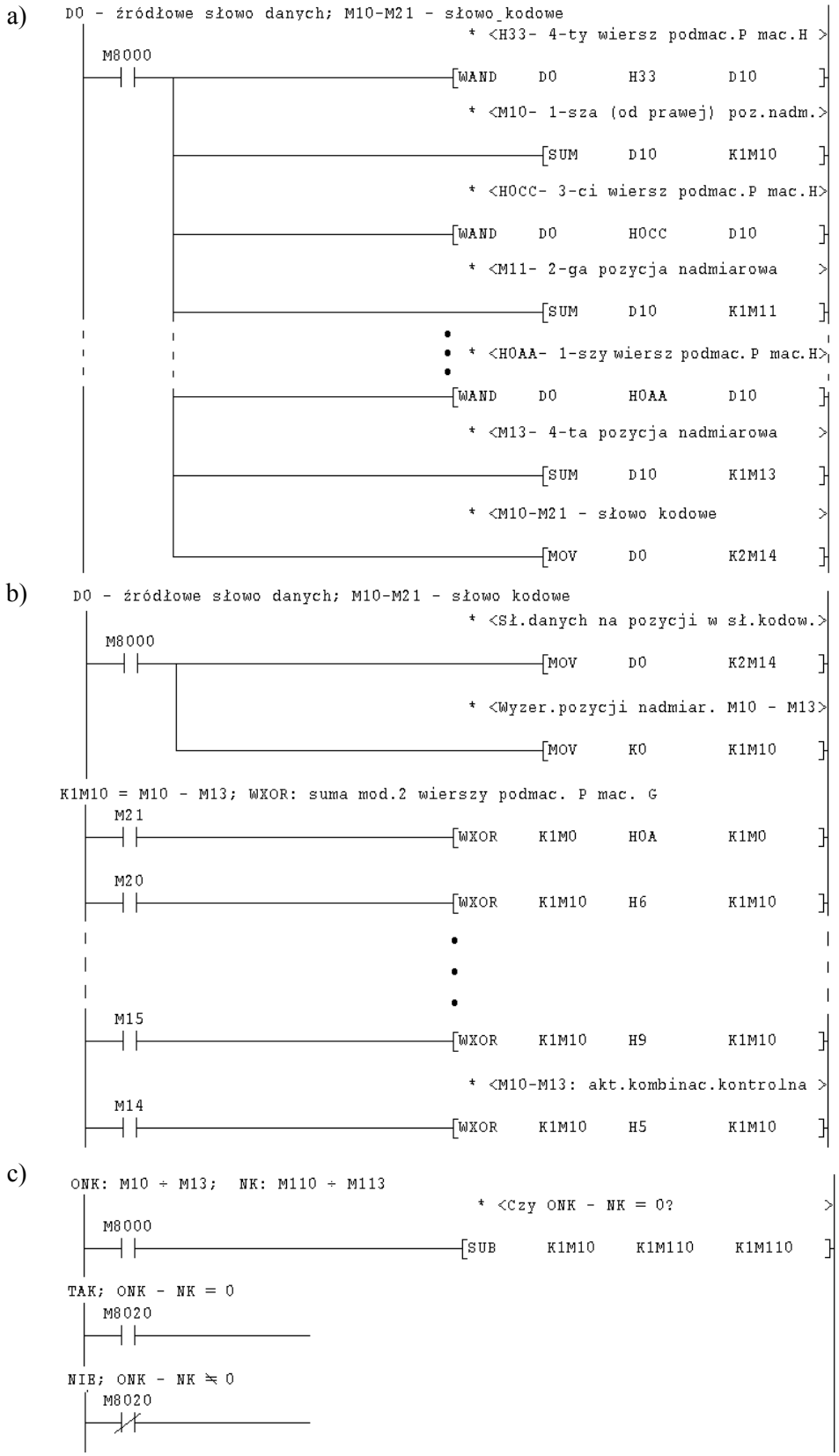
Fragment przykładowego programu drabinkowego, obliczającego słowo kodowe w markerach M21 ÷ M10 opierając się na macierzy H z rys. 2, przedstawiono na rys. 5a. Cztery kolejne instrukcje SUM obliczają kolejne pozycje nadmiarowe, pozostawione kolejno w markerach M10 ÷ M13. Zapis KnMm [2, 4] oznacza rejestr (słowo, liczbę) kolejnych urządzeń bitowych o długości n*4 bitów, rozpoczynający się od markera M o numerze m.

Na rys. 5b przedstawiono fragment programu drabinkowego, obliczającego słowo kodowe na bazie macierzy generującej G z rys. 2. Program działa na podstawie aksjomatu A2 i sumuje mod 2 za pomocą instrukcji WXOR (*Word logical exlusive OR*) 4-pozycyjne słowa nadmiarowe z macierzy G (zapisane w kodzie szesnastkowym – HEX) w 4-bitowym rejestrze markerów M13 ÷ M10. Słowo kodowe uzyskiwane w programie zapisane jest również w rejestrze markerów M21 ÷ M10.

Z porównania programów obu przykładów wynika, że dla przypadku $r = k$ krótszy program uzyskuje się dla metody generacji słowa kodowego, opierając się na macierzy generującej G.

3.2. Kodowanie i dekodowanie tablicowych bloków danych

W większości systemów transmisyjnych jednorazowemu kodowaniu nadmiarowemu podlega większa od jedności liczba bajtów danych. Wówczas mogą być one wbudowane w tablicę o wymiarach n – wierszy i k – kolumn, a każdy wiersz i kolumna mogą być uzupełnione o pozycje nadmiarowe dowolnego kodu. Wówczas otrzymuje się tzw. kod (podwójnie) iterowany (tablica ma dwa wymiary). Kodowanie wierszy tablicy odbywa się zazwyczaj za pomocą tzw. bitu parzystości (lub nieparzystości), który podczas nadawania z nadajnika słowa danych obliczany i dodawany jest układowo na jego końcu w sposób automatyczny. Natomiast kodowanie kolumn odbywa się programowo. W trakcie nadawania kolejnych słów danych (wierszy tablicy) obliczana jest suma wzdłużna w postaci sumy mod 2 poszczególnych pozycji (kolumn) tych słów, która po zakończeniu słów tablicy nadawana jest jako ostatnie słowo bloku kodowego. Słowo sumy wzdłużnej może być również obliczane jako suma arytmetyczna nadawanych słów danych. Metoda ta jest stosowana we wszystkie protokołach wewnętrznych sterowników FX jako zabezpieczenie transmitowanego bloku danych.



Rys. 5. Fragmenty przykładowych programów: (a) kodowania według macierzy H, (b) kodowania według macierzy G, (c) dekodowania

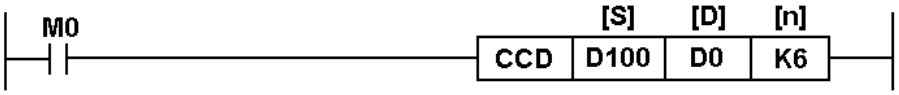
Fig. 5. Snippets of programs used for: (a) encoding with matrix H, (b) encoding with matrix G, (c) decoding

Wprowadzenie bitu kontrolnego do każdego nadawanego słowa danych przez dodatkowe porty szeregowe sterownika FX [1] deklaruje się w ich rejestrach konfiguracyjnych D8120 oraz D8400 i D8420 (FX3U). Natomiast poprawność tego bitu po stronie odbiornika sygnalizowana jest w rejestrach D8063 i D8438 (FX3U).

Obliczenie słowa parzystości wzdłużnej lub sumy kontrolnej można wykonać za pomocą instrukcji WXOR lub ADD, lub też za pomocą jednej specjalizowanej instrukcji CCD (*check code*), której sposób działania prezentują tabele na rys. 6. Stan markera systemowego M8161 definiuje postać bloku danych (również dla procesów nadawania i odbioru), dla którego obliczane są obie sumy. Blok ten składać się może albo z [*n*/2] słów sterownika dla M8161 = „0”, albo z młodszych bajtów kolejnych [*n*] słów bloku dla wartości M8161 = „1”. Po wykonaniu instrukcji CCD wybrany rejestr wyniku jest nadawany za blokiem danych.

Dekodowanie odebranego bloku kodowego jest analogiczne do odebranego słowa kodowego. Najpierw na bajtach bloku danych należy wykonać instrukcję CCD określając ONK, a następnie należy porównać go z odebranym nadmiarem kodowym bloku NK za po-mocą podobnego programu jak na rys. 3c.

CCD: obliczenie sumy arytmetycznej i sumy parzystości bajtów w rejestrach celu [D] i [D+1] ze stosu [n] słów o adresie początkowym [S]



M8161 = „0”									
Źródło (S)			Kombinacja bajtu							
D100	H	FF	1	1	1	1	1	1	1	1
	L	FF	1	1	1	1	1	1	1	1
D101	H	FF	1	1	1	1	1	1	1	1
	L	00	0	0	0	0	0	0	0	0
D102	H	F0	1	1	1	1	0	0	0	0
	L	0F	0	0	0	0	1	1	1	1
Parzystość pionowa: D1			0	0	0	0	0	0	0	0
Suma: D0			3FC							

M8161 = „1”									
Źródło (S)		Kombinacja bajtu							
D100 L	FF	1	1	1	1	1	1	1	1
D101 L	00	0	0	0	0	0	0	0	0
D102 L	0F	0	0	0	0	1	1	1	1
D103 L	F0	1	1	1	1	0	0	0	0
D104 L	F0	1	1	1	1	0	0	0	0
D105 L	0F	0	0	0	0	1	1	1	1
Parzystość pionowa: D1		1	1	1	1	1	1	1	1
Suma: D0		2FD							

Rys. 6. Instrukcja CCD sterownika FX obliczająca sumę wzdłużną
Fig. 6. FX controller CCD instruction calculating the longitudinal sum

3.3. Kodowanie i dekodowanie wielomianowe

W zabezpieczaniu danych najszersze zastosowanie znalazły kody w zapisie wielomianowym. Uzyskuje się je po wprowadzeniu operacji mnożenia binarnego do grupy słów kodowych, przez co otrzymuje się strukturę algebraiczną zwaną pierścieniem. Struktura ta ma wszystkie atrybuty grupy, lecz elementy zbioru grupy (słowa kodowe) muszą spełniać dodatkowo tę nowo wprowadzoną własność, czyli być podzielne przez określony czynnik. Wprowadzenie operacji mnożenia do zbioru słów kodowych wiąże się z przedstawieniem tych

słów w postaci wielomianu o współczynnikach binarnych. Wartość tych współczynników stanowi kombinację słowa kodowego. Poniżej przedstawiono konwersję zapisu przykładowego słowa 4-bitowego w postać wielomianową, gdzie znak $\oplus$ jest operatorem mod 2.

$$H = 1011 \Leftrightarrow h(x) = 1 \times x^3 \otimes 0 \times x^2 \otimes 1 \times x^1 \otimes 1 \times x^0 = x^3 \otimes x \otimes 1 \tag{2}$$

a)

x^7	x^6	x^5	x^4	x^3	x^2	x	1	<= Pozycje dzielnej	Składnik ilorazu h(x)
x^7	x^6	x^5	x^4	0	0	x	1	Dzielna	
x^7	0	x^5	x^4	0	0	0	0	s(x) : g(x)	x^4
0	x^6	0	0	0	0	x	1	$= s_1(x)$	
	x6	0	x^4	x^3	0	0	0	$s_1(x) : g(x)$	x^3
	0	0	x^4	x^3	0	x	1	$= s_2(x)$	
			x^4	0	x^2	x	0	$s_2(x) : g(x)$	x
			0	x^3	x^2	0	1	$= s_3(x)$	
				x^3	0	x	1	$s_3(x) : g(x)$	1
				0	x^2	x	0	= r(x)	

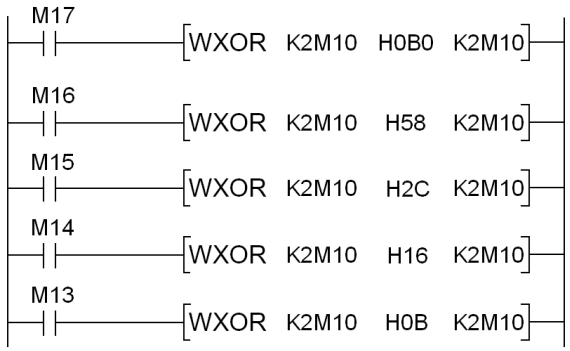
b)

	x^7	x^6	x^5	x^4	x^3	x^2	x	1	<= Pozycje dzielnej	Składnik ilorazu h(x)
	M17	M16	M15	M14	M13	M12	M11	M10	<= Markery Mi dzielnej	
	x^7	x^6	x^5	x^4	0	0	x	1	Dzielna s(x)	
WXOR	1	1	1	1	0	0	1	1	Stan Mi	
	1	0	1	1	0	0	0	0	s(x) : H0B0	x^4=M17=1
= WXOR	0	1	0	0	0	0	1	1	$= s_1(x)$	
		1	0	1	1	0	0	0	$s_1(x)$: H58	x^3=M16=1
= WXOR		0	0	1	1	0	1	1	$= s_2(x)$	
				1	0	1	1	0	$s_2(x)$: H16	x^1=M14=1
= WXOR				0	1	1	0	1	$= s_3(x)$	
					1	0	1	1	$s_3(x)$: H0B	x^0=M13=1
=					0	1	1	0	= r(x)	

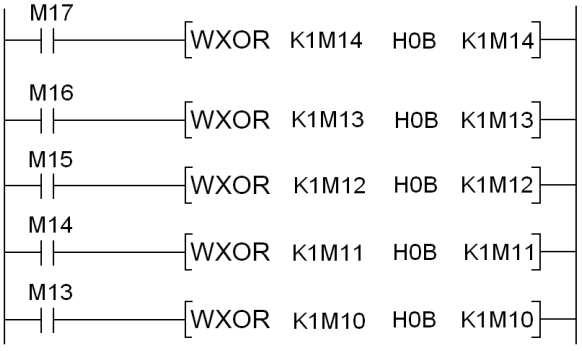
c)

	x^7	x^6	x^5	x^4	x^3	x^2	x	1	<= Pozycje dzielnej	Składnik ilorazu h(x)
	M17	M16	M15	M14	M13	M12	M11	M10	<= Markery Mi dzielnej	
	x^7	x^6	x^5	x^4	0	0	x	1	Dzielna s(x)	
WXOR	1	1	1	1	0	0	1	1	Stan Mi	
	1	0	1	1	0	0	0	0	s(x) : H0B	x^4=M17= 1
= WXOR	0	1	0	0	0	0	1	1	$= s_1(x)$	
		1	0	1	1	0	0	0	$s_1(x)$: H0B	x^3=M16=1
= WXOR		0	0	1	1	0	1	1	$= s_2(x)$	
				1	0	1	1	0	$s_2(x)$: H0B	x^1=M14= 1
= WXOR				0	1	1	0	1	$= s_3(x)$	
					1	0	1	1	$s_3(x)$: H0B	x^0=M13= 1
=					0	1	1	0	= r(x)	

d)



e)



Rys. 7. Dzielenie dwóch wielomianów: (a) zapis operacyjny, (b)-(e) operacja na markerach

Fig. 7. Dividing two polynomials: (a) operational notation, (b)-(e) operation on markers

Do wygenerowania kodu o parametrach (n, k) należy określić wielomian stopnia $r = n - k$, który nazywany jest wielomianem generującym $G(x)$. Wielomian ten reprezentuje jedno wybrane słowo kodowe, na bazie którego można wygenerować wszystkie pozostałe słowa kodowe. Każdy wielomian kodowy s(x) jest podzielny bez reszty przez wielomian generujący, co można określić następującym równaniem:

$$s(x) \bmod g(x) = 0 \tag{3}$$

W procesie kodowania wielomianowego dla kodu rozdzielnego podstawową rolę odgrywa operacja dzielenia wielomianów Na rys. 7a przedstawiono przykładowy proces dzielenia dwóch wielomianów: dzielnej: $s(x) = x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x \oplus 1$ i dzielnika: $g(x) = x^3 \oplus x \oplus 1$.

Przechodząc do realizacji programowej dzielenia powyższych wielomianów w sterowniku FX, należy najpierw wpisać wielomian dzielnej $s(x)$ do rejestru kolejnych markerów M10÷M17, co przedstawiają tabele na rys. 7b i 7c. Oba rysunki przedstawiają dwie metody dzielenia. W pierwszej z nich (rys. 7b) dzielnik jest równy zmiennej długości dzielnej dla każdego etapu dzielenia (sumowania mod 2), a jego najmłodsza pozycja jest zawsze w markerze M10. W drugiej (rys. 7c) dzielnik ma stałą swoją pierwotną długość, lecz przypisywany jest od kolejnego młodszego markera w kolejnym etapie dzielenia – sumowania mod 2 z dzielną.

Obie metody mają identyczną strukturę programu sterownika, co przedstawiono na rys. 7d dla metody z dzielnikiem o zmiennej długości i na rys. 7e dla metody z dzielnikiem o stałej długości. W pierwszej z nich każda warunkowa instrukcja WXOR ma inny drugi operand źródłowy (dzielnik zapisany w kodzie HEX), natomiast w drugiej operand ten ma stałą kombinację, a zmienia się tylko zakres markerów, z którymi jest on sumowany mod 2. Warunkiem wstępnym wykonania danego programu jest wpisanie dzielnej do rejestru markerów M10÷M17. Efektem każdego programu jest reszta z dzielenia, znajdująca się w markerach M10÷M12.

Proces kodowania wielomianowego dla kodu rozdzielnego polega na takim określeniu kombinacji pozycji nadmiarowych słowa kodowego, aby słowo to było podzielne bez reszty przez wielomian generujący zgodnie z zależnością (2). Można wykazać [4], że tą kombinacją jest reszta z dzielenia słowa danych, pomnożonego przez czynnik x^r (przesuniętego w lewo o r pozycji). To przesunięcie powoduje wpisanie ciągu zer na pozycje nadmiarowe, w które po operacji dzielenia wpisywana jest obliczana reszta.

W wyniku dzielenia otrzymuje się wielomian ilorazu (wynik dzielenia), który w procesie kodowania jest nieistotny oraz resztę $r(x)$, która jest dalej wykorzystywana.

Programy przytoczone na rys. 7d i 7e służą do tworzenia słów kodowych dla kodu o parametrach (n, k) = (8, 5), z wielomianem generującym $g(x) = x^3 \oplus x \oplus 1$. Aby wygenerować słowo kodowe tego kodu, należy 5-bitowe słowo danych źródłowych $h(x)$ wpisać do marke-

rów M13÷M17 (przy zerowych wartościach M10÷M12) i wykonać daną wersję programu. Poszukiwana reszta pozostanie w markerach M10÷M12. Następnie należy ponownie wpisać słowo danych $h(x)$ do markerów M13÷M17, przez co w rejestrze markerów M10÷M17 uzyskuje się poszukiwane słowo kodowe.

3.4. Kodowanie wielomianowe w sterowniku FX3U

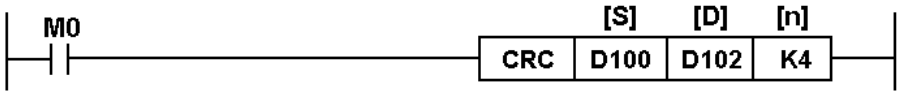
Przykładowe programy z rys. 7d i 7e są również wzorcem dla tworzenia programów kodowania wielomianowego dla innych parametrach kodu (n, k), szczególnie dla k=i*8 i mogą być zastosowane w każdym typie sterowników FX. W najszybszym ze sterowników FX o symbolu FX3U wprowadzono specjalizowaną instrukcję o symbolu CRC obliczania nadmiaru kodowego, w postaci reszty z dzielenia przez jeden ze znormalizowanych wielomianów generujących o symbolu CRC16, który ma następującą postać:

$$CRC16 = x^{16} \otimes x^{15} \otimes x^{2} \otimes 1 \tag{4}$$

Instrukcja CRC (*cyclic redundancy check*) wykonuje dzielenie na zadeklarowanym bloku danych (analogicznie do instrukcji CDC) i obliczoną 16-bitową resztę wpisuje do wyspecyfikowanego rejestru. Zapis instrukcji na 4-bajtowym bloku danych dla operacji na całych słowach (M8161 = „0”) przedstawiono na rys. 8. Działanie instrukcji dla tego przypadku obrazuje górna część tabeli danych źródłowych i wyniku CRC.

Dla przypadku wartości markera M8161 = „1” blok danych określony jest przez młodsze bajty zadeklarowanych słów bufora źródłowego, stąd w deklaracji instrukcji dla tego przypadku przedstawionym w tablicy na rys. 8, rejestr wyniku musiałby mieć zapis [D] = D104. Młodszy bajt obliczanej reszty umieszczany jest w rejestrze następnym D105.

CRC: obliczanie 16-bitowego wielomianowego nadmiaru kodowego w rejestrze celu [D] jako reszty z dzielenia wielomianu danych zawartych w stosie [n] bajtów o adresie początkowym [S] przez wielomian generujący CRC16 = $x^{16} + x^{15} + x^{2} + 1$



Źródło (S)		Bajt starszy H								Bajt młodszy L								
Pozycja Di;	i:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Składnik x^{m+j};	j:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	M8161 = „0"
D100	m=32	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	
D101	m=16	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	1	
CRC:D102	m= 0	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	
Składnik x^{m+j};	j:									0	1	2	3	4	5	6	7	
D100	m=40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
D101	m=32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
D102	m=24	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	M8161 = „1"
D103	m=16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
CRC_H:D104	m= 8	0	0	0	0	0	0	0	0	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	
CRC_L:D105	m= 0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	

Rys. 8. Instrukcja CRC i tabela z danymi źródłowymi i obliczonym nadmiarem kodowym
Fig. 8. CRC instruction and a table containing the source data and the calculated code redundancy

Taki sposób deklaracji umieszczenia wyniku dzielenia zaraz za blokiem danych źródłowych pozwala na wykorzystanie rejestrów używanych przez instrukcję CRC również jako bufora nadawczego.

W tabeli na rys. 8 pokazano numerację pozycji (bitów) poszczególnych bajtów słowa kodowego w zapisie wielomianowym, czyli określenie wartości wykładnika $m+j$ poszczególnych składników x^{m+j} wielomianu kodowego. Przyjęto tu zasadę, że pierwszy nadawany bit (d0) bajtu danych w transmisji szeregowej jest najstarszą pozycją (x^7 dla bajtu) w zapisie wielomianowym. Miałoby to istotne znaczenie w odbiorniku transmisyjnym, w którym zastosowano by dekoder szeregowy, dzielący odbierany strumień danych przez wielomian CRC16. Aby upewnić się, że instrukcja CRC generuje resztę wg wielomianu CRC16 zgodnie z regułą, która miałby być zaimplementowana w układzie projektowanego odbiornika (poza systemem sterownika FX3U), sprawdzić należy sposób generacji tej reszty. Jeśli założyć, że słowo danych będzie słowem 16-bitowym, to instrukcja CRC ma za zadanie określenie nadmiaru kodowego dla kodu $(n, k) = (32, 16)$. Jeśli w takim kodzie założyć, że pozycja $x^{16} = 1$, to nadmiar kodowy (reszta z dzielenia) powinien mieć pozostałe składniki wielomianu generującego CRC16, czyli $r(x) = x^{15} \oplus x^2 \oplus 1$.

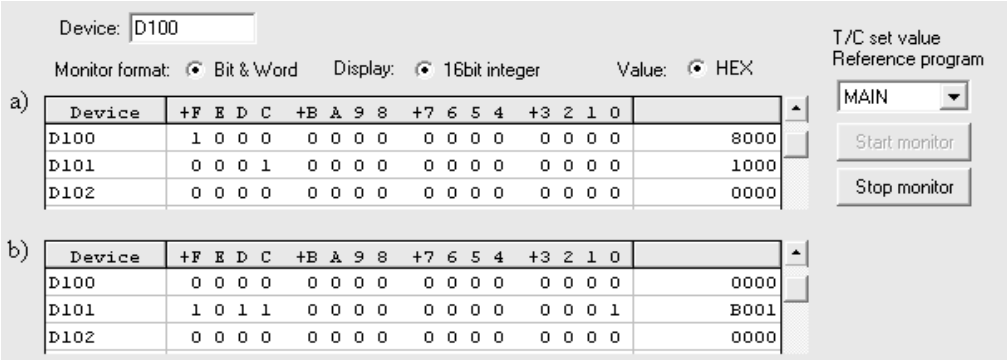
Na rys. 9 w tabeli (a) pokazano wynik takiego eksperymentu (rysunek przedstawia fragmenty okna monitora urządzeń oprogramowania narzędziowego GX-Developer sterowników FX), w którym w instrukcji CRC zadeklarowano liczbę bajtów danych $k = 2$; bajty rejestru D100. Pozycja x^{16} słowa danych w rejestrze D100 jest równa „1” (pozycja d15; patrz tabela (a) na rys. 7), natomiast resztę z dzielenia przedstawia stan rejestru D101. Pozycja „1” w tym rejestrze odpowiada składnikowi x3 wielomianu reszty, co jest niezgodne z kombinacją oczekiwaną. W wyniku tego kodowania otrzymano wielomian $y1(x) = x^{16} \oplus x^3$.

Zachodzi pytanie, czy kod generowany przez instrukcję CRC jest kodem liniowym w myśl zapisu (1), czyli czy spełniony jest aksjomat A1 z podrozdziału 2. W tym celu zadeklarowano wykonanie tego samego programu z zerowym słowem danych w rejestrze D100. Wynik rejestracji obliczonej reszty przez instrukcję CRC przedstawiono w rejestrze D101 w tabeli (b) na rys. 9. Zamiast wyniku zerowego otrzymano wielomian $r1(x) = x^{15} \oplus x^3 \oplus x^2 \oplus 1$, co sugeruje, że kod generowany za pomocą instrukcji CRC nie jest kodem liniowym.

Jedyną alternatywą mogącą zaprzeczyć temu stwierdzeniu jest przyjęcie założenia, że kod generowany przez instrukcję CRC jest kodem warstwowym kodu liniowego. Kod warstwowy otrzymuje się z kodu liniowego przez dodanie (mod 2), do każdego słowa kodowego, słowa o stałej kombinacji, tzw. słowa tworzącego warstwę. Tym słowem może być wielomian niezerowej reszty $r1(x)$, otrzymanej w wyniku dzielenia zerowego słowa danych. Dodanie tego wielomianu do określonego wyżej wielomianu kodowego $y1(x)$ daje następujący wynik:

$$y1(x) \otimes r1(x) = (x^{16} \otimes x^3) \otimes (x^{15} \otimes x^3 \otimes x^2 \otimes 1) = x^{16} \otimes x^{15} \otimes x^2 \otimes 1 = CRC16 \qquad (4)$$

Uzyskanie, w wyniku powyższego rozumowania, wielomianu generującego CRC16 potwierdza jednak poprawność deklaracji jego stosowania w instrukcji CRC.



Rys. 9. Tabele rejestracji wyników instrukcji CRC dla wielomianów danych $s(x) = 1$ i $s(x) = 0$
Fig. 9. Tables containing the output of CRC instructions for the data polynomials $s(x) = 1$ and $s(x) = 0$

4. Podsumowanie

Z przytoczonych przykładów w podrozdziale 3 wynika, że w każdym typie sterownika z rodziny FX można zastosować dowolny sposób kodowania i dekodowania nadmiarowego danych. Nawet, jeśli sterownik nie ma specjalizowanej do tego celu instrukcji (np. CRC), to można zastąpić ją procedurą bazującą na instrukcjach uniwersalnych. Tworzenie takiej procedury jest procesem bardziej skomplikowanym i wymagającym znacznej wiedzy od programisty, ale jest to zawsze możliwe. Natomiast, jeśli w systemie sieciowym występują stacje ze sterownikami z grupy FX3U, to w stacjach tego systemu wskazane byłoby zastosowanie sposobu kodowania opartego na instrukcji CRC (wielomianie CRC16), nawet jeśli w pozostałych stacjach należałoby wprowadzić programowe procedury, pozwalające na generowanie nadmiaru, wg reguły stosowanej w instrukcji CRC.

BIBLIOGRAFIA

1. Mitsubishi Electric: MELSEC FX Series. User's Manual (Data Communication) FX1S/FX1N/FX2N(C)/FX3U Interface Modules. 2007.
2. Mitsubishi Electric: PROGRAMMING MANUAL – Basic & Applied Instructions Edition. FX3u/FX3UC Series Programmable Controllers. 2005.
3. Kubiak Z., Mielcarek R.: Problematyka kodu optymalnego dla zabezpieczenia danych transmisyjnych w Radiotelefonicznym Systemie Telemechaniki. Prace II MKNT pt.: REAL-TIME SYSTEMS '95, str. 394-402, Szklarska Poręba, wrzesień 1995.

4. Dróżdż J.: Podstawy kodowania nadmiarowego. WPP 1980.
5. Mielcarek R.: Programowanie Sterowników PLC – Przewodnik do ćwiczeń laboratoryjnych. WPP 2012.

Wpłynęło do Redakcji 12 marca 2013 r.

Abstract

The paper presents software capabilities to perform redundant data coding in Mitsubishi Electric FX PLC programmable controllers. There are two instances where this problem is particularly significant. The first situation is when a programmer's own transmission protocol is used. The second one is the case when there is a need for software verification of the protocol frame received, and the controller is not equipped with specialized decoding instructions.

The paper consists of 4 sections. Section 1 briefly introduces the subject. Section 2 discusses the encoding (Fig. 1) and decoding (Fig. 3) processes in a systematic block code defined with: a system of linear equations (1), a generator matrix G and a control matrix H (Fig. 2). Section 3 is the main part of the paper discussing the process of software encoding and decoding in an FX controller. In Subsection 3.1 we consider a problem of data encoding based on matrices H and G (Fig. 4 and 5), for the code with parameters (n, k). Subsection 3.2 examines the problem of data blocks encoding using the longitudinal sum (Fig. 6). Subsection 3.3 discusses the software approach to polynomial notation encoding using a special WXOR instruction (Fig. 7). Subsection 3.4 discusses the use of specialized controller instruction CRC that encodes data blocks using the standard generator polynomial CRC16. Figure 8 depicts the use of this instruction for encoding the entire 16-bit system words of a controller (the system marker M8161 = 0), as well as for encoding only the least significant bytes of these words (M8161 = 1). In the second part of Subsection 3.4, the data encoding using this instruction is verified. It is proved that the encoding process is performed using the layer encoder (Fig. 9), with the layer word generated with a polynomial $r1(x) = x^{15} \oplus x^3 \oplus x^2 \oplus 1$. Section 4 concludes the paper.

Adres

Roman MIELCAREK: Politechnika Poznańska, Instytut Informatyki, ul. Piotrowo 2, 60-965 Poznañ, Polska, roman.mielcarek@cs.put.poznan.pl