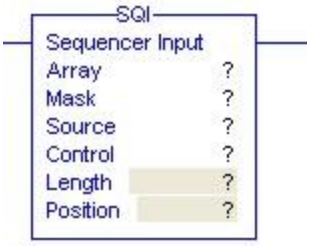
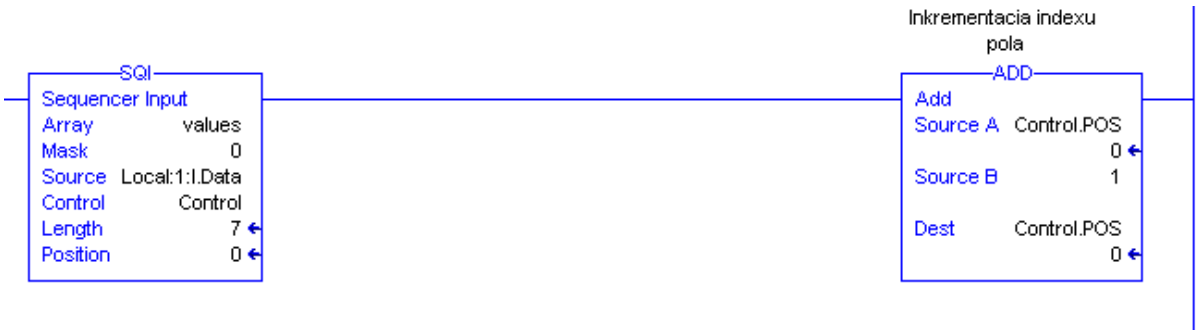
Inštrukcie sekvencií



SQI (Sekvencer vstupu)

SQI je inštrukcia, ktorá sa vykoná, ak stav rungu je priechodný. Táto inštrukcia sa väčšinou používa párovo s inštrukciou SQO. SQI inštrukcia vykonáva porovnávanie vstupných dát cez masku s dátami uloženými v poli . Vstupných dát sa adresujú Source slovom. V Array slove je uložene pole dát, ktoré porovnávame zo vstupnými dátami. Do Control slova sa zadáva adresa ovládacej štruktúry pre inštrukciu. Do Position slova sa ukladá počiatočná pozícia poľa dát, od ktorej chceme porovnávať a slovo Length udáva koľko prvkov poľa prečítame od počiatočnej pozícii, čiže počet prvkov poľa. Inštrukcia SQI pracuje na súvislej pamäti. Tato inštrukcia nezabezpečuje inkrementáciu indexu poľa, preto sa používa párovo s inštrukciou SQO. Ak by sme nechceli používať inštrukciu SQO, ale samostatnú inštrukciu SQI, potrebujeme zabezpečiť inkrementaciu indexu pola pomocou inštrukcie ADD.



16-bitová adresa ovládacej štruktúry SQI inštrukcie sa skladá:

Ovládacia štruktúra:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	číslo bitu
	—					N	N	N	N	N	N	N	N	N	N	N	
Slovo 0	-	-	-	-	ER	- N	- N	- N	- N	- N	- N	- N	- N	- N	- N	- N	
Slovo 1	**Dĺžka sekvenčného súboru (Lenght)**																
Slovo 2	**Pozícia (Position)**																

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	číslo bitu

ER Bit, (Error bit), Bit chyby. Nastaví sa, ak hodnoty v slovách Position a Length sú menšie ako 0, alebo ak hodnota v Position slove je väčšia ako hodnota v Length slove.

Sekvenčné slovo 1:

Lenght udáva počet prvkov v sekvenčnom poli.

Sekvenčné slovo 2:

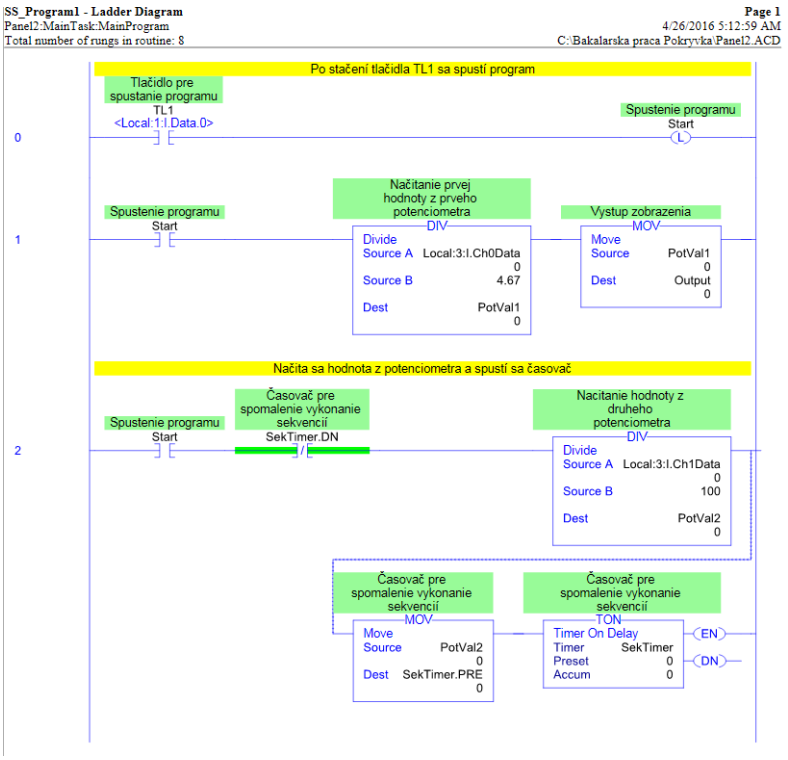
Position udáva počiatočný index poľa, od ktorého prvku sa majú dáta porovnávať.

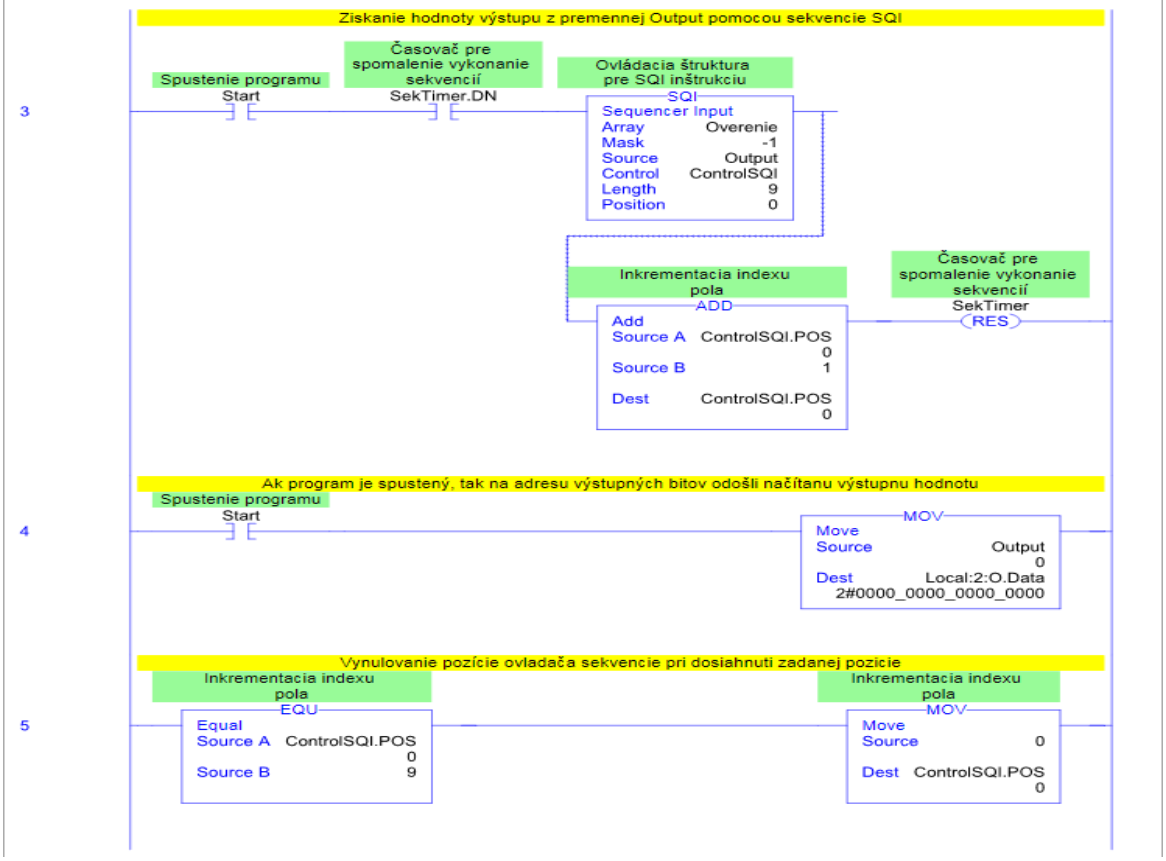
SS_Program1

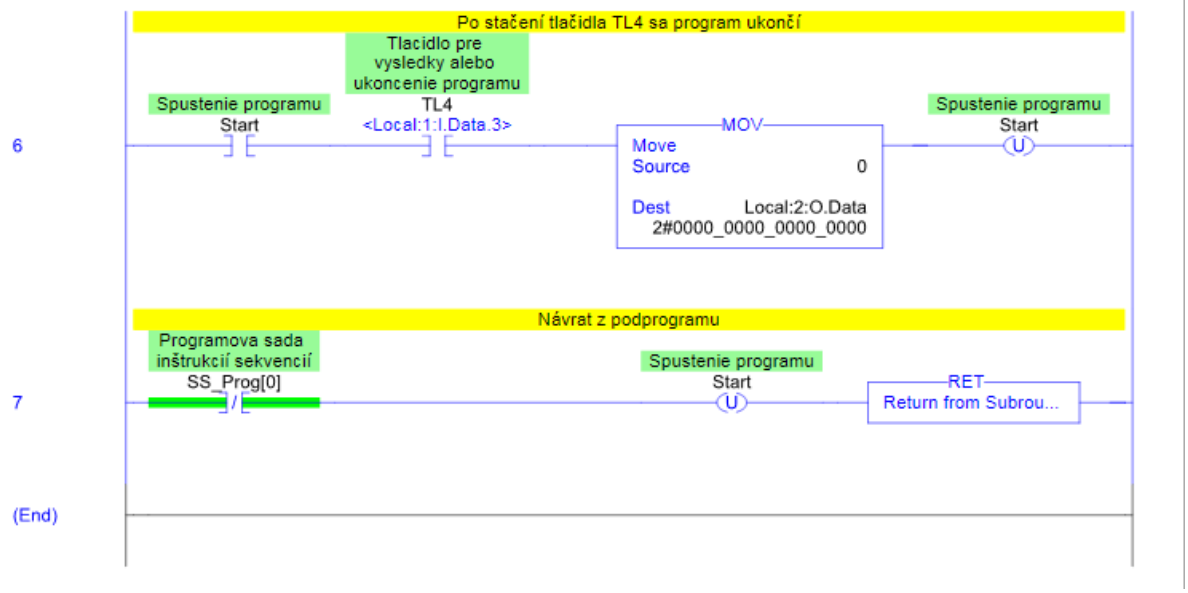
Úloha: Úlohou programu je znázornenie funkcionality inštrukcie SQI, ktorá overuje načítané dáta z dátami v zadanom poli.

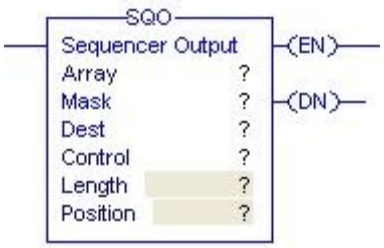
Riešenie: Úlohu bude realizovaná nasledovne. Pomocou tlačidla TL1 sa spustí program. Po spustení programu sa načíta hodnota z prvého potenciometra, ktorá bude predstavovať vstup do sekvencie SQI a taktiež sa načíta druha hodnota z druhého potenciometra, ktorá nastaví dĺžku počítania časovača, ktorý určuje rýchlosť vykonávania inštrukcie SQI. Ak časovač dopočíta vykoná sa inštrukcia SQI, ktorá porovná načítanú hodnotu z hodnotami v poli a ak sa táto hodnota zhoduje aspoň z jednou hodnotou v poli, tak sa odošle na adresu výstupných bitov. Po stlačení tlačidla TL4 sa program zastaví. Do podprogramu SS_Program1 vstúpime, ak premenná SS_Prog[0] nadobúda logickú jednotku.

Rebríková schéma podprogramu SS_Program1:




Ziskanie hodnoty výstupu z premennej Output pomocou sekvencie SQI
3
Spustenie programu
Start
Časovač pre spomalenie vykonanie sekvencií
SekTimer.DN
Ovládacia štruktura pre SQI inštrukciu
SQI
Sequencer Input
Array Overenie
Mask -1
Source Output
Control ControlSQI
Length 9
Position 0
Inkrementacia indexu pola
ADD
Add
Source A ControlSQI.POS 0
Source B 1
Dest ControlSQI.POS 0
Časovač pre spomalenie vykonanie sekvencií
SekTimer
RES
Ak program je spustený, tak na adresu výstupných bitov odošli načítanu výstupnu hodnotu
4
Spustenie programu
Start
MOV
Move
Source Output 0
Dest Local:2:O.Data 2#0000_0000_0000_0000
Vynulovanie pozície ovladača sekvencie pri dosiahnuti zadanej pozicie
5
Inkrementacia indexu pola
EQU
Equal
Source A ControlSQI.POS 0
Source B 9
Inkrementacia indexu pola
MOV
Move
Source 0
Dest ControlSQI.POS 0


Po stačení tlačidla TL4 sa program ukončí
6
Spustenie programu
Start
Tlacidlo pre vysledky alebo ukoncenie programu
TL4
<Local:1:I.Data.3>
MOV
Move
Source 0
Dest Local:2:O.Data 2#0000_0000_0000_0000
Spustenie programu
Start
U
Návrat z podprogramu
7
Programova sada inštrukcií sekvencií
SS_Prog[0]
Spustenie programu
Start
U
RET
Return from Subrou...
(End)



SQO (Sekvencer výstupu)

SQO je inštrukcia, ktorá sa vykoná, ak stav rungu je priechodný. Táto inštrukcia sa väčšinou používa párovo s inštrukciou SQI. SQO inštrukcia vykonáva cyklické menenie pozície poľa výstupných dát, presunutie dát cez masku na danú pozíciu a výsledok uloží do cieľovej adresy. Pole výstupných dát je uložene v Array slove. Výsledok sa ukladá do Dest slova, ktoré väčšinou predstavuje výstupná adresa. Do Control slova sa zadáva adresa ovládacej štruktúry pre inštrukciu. Do Position slova sa ukladá počiatočná pozícia výstupného poľa a slovo Length udáva koľko prvkov poľa prečítame od počiatočnej pozícii, čiže počet prvkov poľa. Inštrukcia SQO pracuje na súvislej pamäti.

16bitová adresa ovládacej štruktúry SQO inštrukcie sa skladá:

Ovládacia štruktúra:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	číslo bitu
						N	N	N	N	N	N	N	N	N	N	N	
Slovo 0	**EN**	-	**DN**	-	**ER**	- N	- N	- N	- N	- N	- N	- N	- N	- N	- N	- N	
Slovo 1	**Dĺžka sekvenčného súboru (Lenght)**																
Slovo 2	**Pozícia (Position)**																

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	číslo bitu

ER Bit, (Error bit), Bit chyby. Nastaví sa, ak hodnoty v slovách Position a Length sú menšie ako 0, alebo ak hodnota v Position slove je väčšia ako hodnota v Length slove.

DN Bit, (Done bit), Bit ukončenia sa nastaví, ak všetky prvky boli odoslané na adresu Dest slova.

EN Bit, (Enable bit), Bit povolenia sa nastaví, ak stav rungu je priechodný. Povoľuje vykonávanie inštrukcie.

Sekvenčné slovo 1:

Lenght udáva počet prvkov v sekvenčnom poli.

Sekvenčné slovo 2:

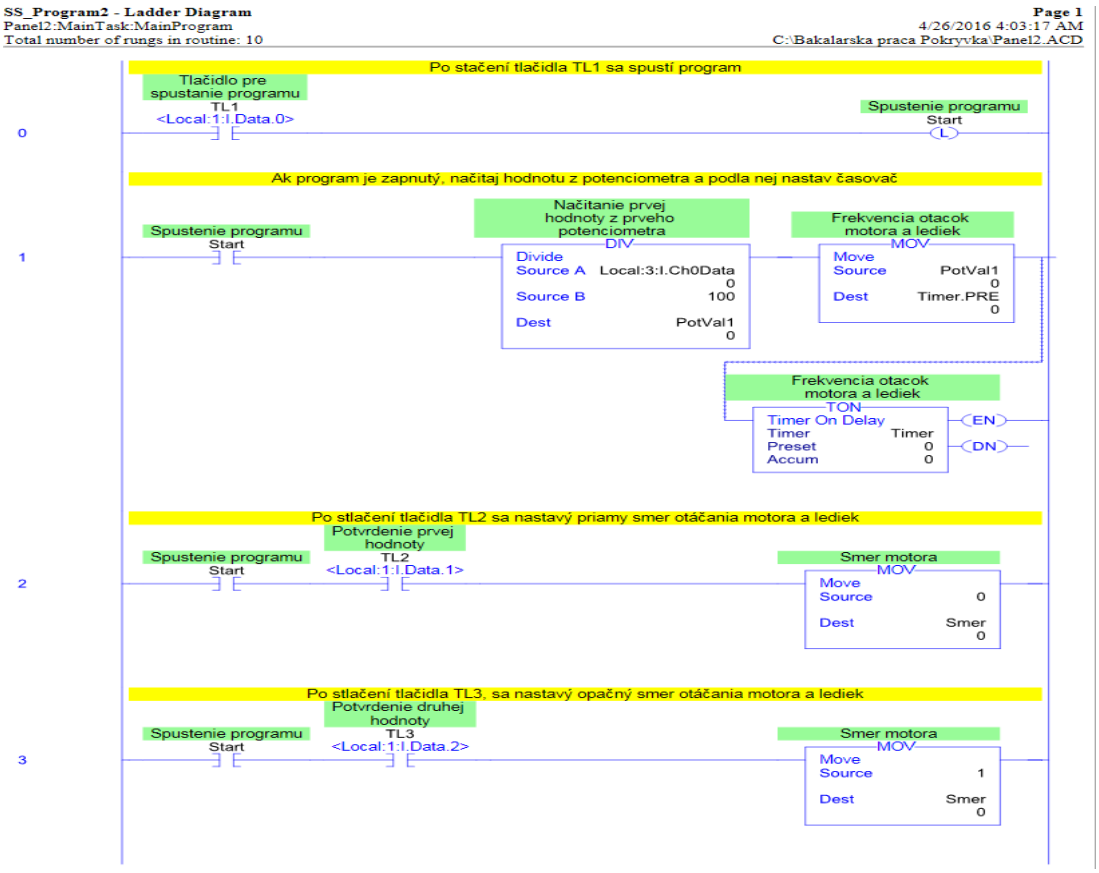
Position udáva počiatočný index poľa, od ktorého prvku sa ma index inkrementovať.

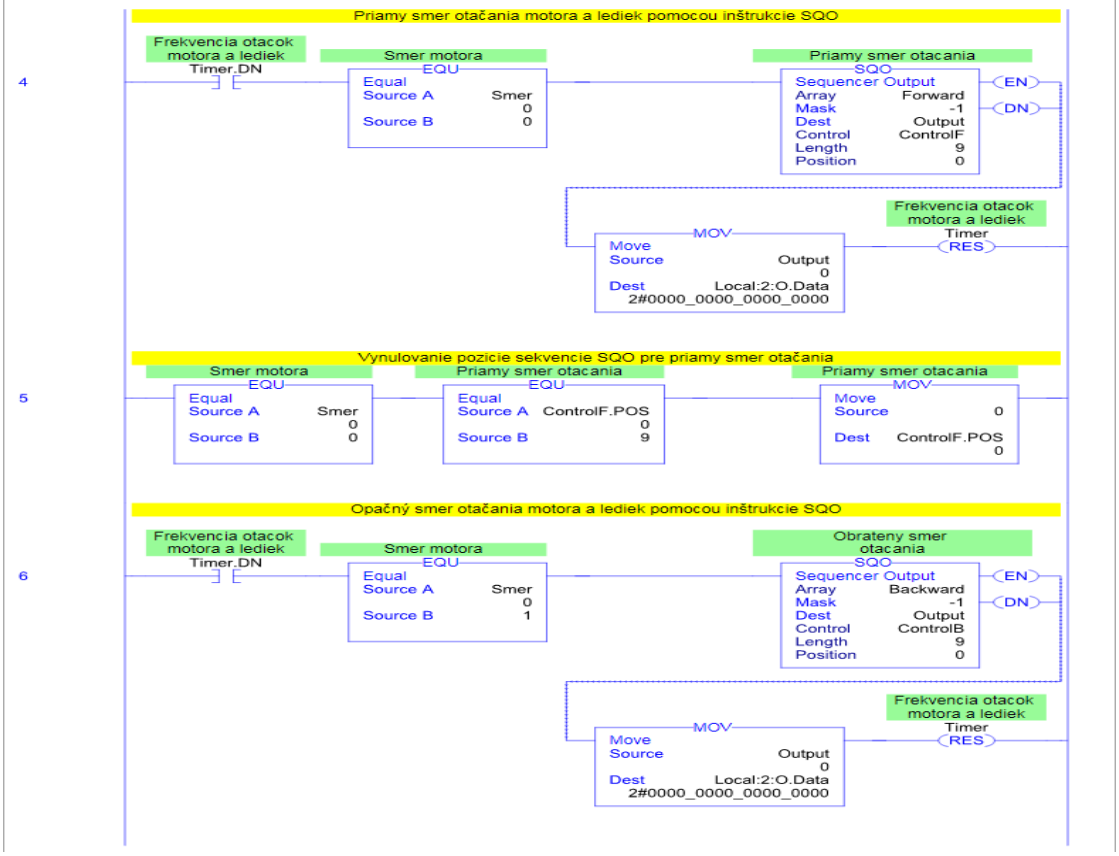
SS_Program2

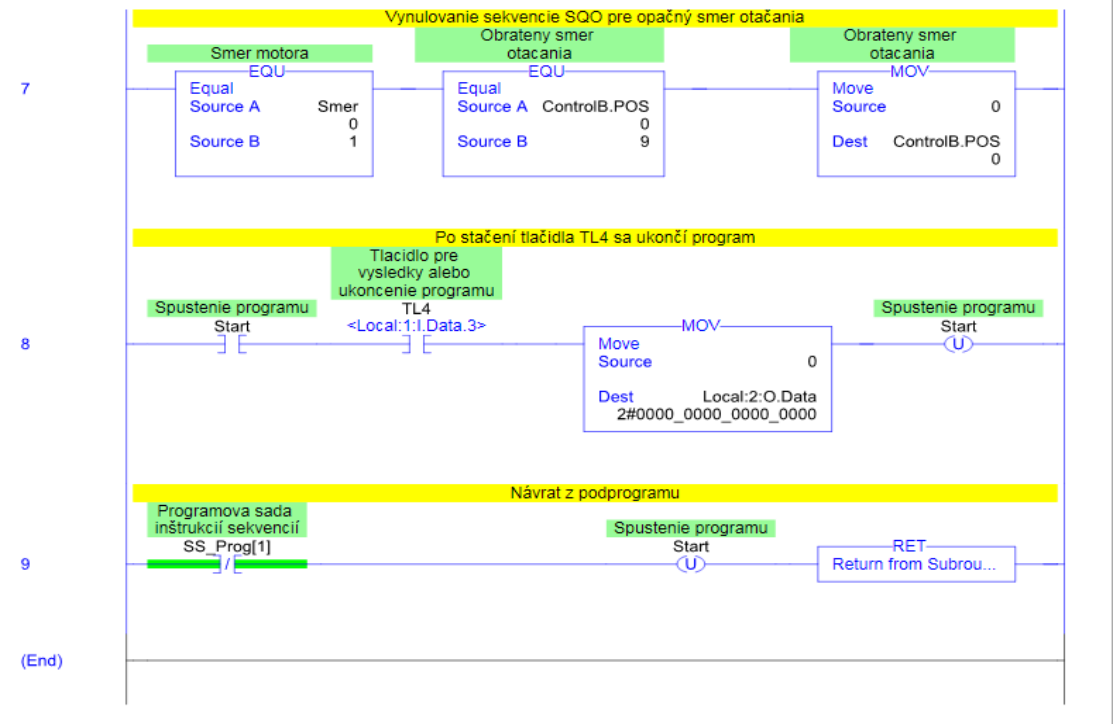
Úloha: Úlohou programu je znázornenie funkcionality inštrukcie SQO, ktorá vkladá hodnoty zadaného poľa do premennej adresovanej Dest slovom.

Riešenie: Úlohu bude realizovaná na príklade odtáčania krokového motora a led hada. Pomocou tlačidla TL1 sa spustí program. Po spustení programu sa načíta hodnota z prvého potenciometra, ktorá nastaví dĺžku počítania časovača, ktorý určuje rýchlosť vykonávania inštrukcie SQO. Nasledovne sa určí smer otáčania motora a led hada Predvolený smer otáčania je v smere hodinových ručičiek, pričom smer otáčania volíme podľa tlačidiel TL2 a TL3. Tlačidlo TL2 nastaví priamy smer otáčania a tlačidlo TL3 nastaví spätný smer otáčania. Ak časovač dopočíta vykoná sa zvolená inštrukcia SQO, podľa smeru otáčania, keďže priamy smer otáčania predstavuje rozdielne pole hodnôt, ako spätný smer otáčania. Po stlačení tlačidla TL4 sa program zastaví. Do podprogramu SS_Program2 vstúpime, ak premenná SS_Prog[1] nadobúda logickú jednotku.

Rebríková schéma podprogramu SS_Program2:




Priamy smer otačania motora a lediek pomocou inštrukcie SQO
Frekvencia otacok motora a lediek Timer.DN
Smer motora
EQU
Equal
Source A Smer 0
Source B 0
Priamy smer otacania
SQO
Sequencer Output
Array Forward
Mask -1
Dest Output
Control ControlF
Length 9
Position 0
EN
DN
Frekvencia otacok motora a lediek Timer
RES
MOV
Move
Source Output 0
Dest Local:2:O.Data 2#0000_0000_0000_0000
4
Vynulovanie pozicie sekvencie SQO pre priamy smer otačania
Smer motora
EQU
Equal
Source A Smer 0
Source B 0
Priamy smer otacania
EQU
Equal
Source A ControlF.POS 0
Source B 9
Priamy smer otacania
MOV
Move
Source 0
Dest ControlF.POS 0
5
Opačný smer otačania motora a lediek pomocou inštrukcie SQO
Frekvencia otacok motora a lediek Timer.DN
Smer motora
EQU
Equal
Source A Smer 0
Source B 1
Obrateny smer otacania
SQO
Sequencer Output
Array Backward
Mask -1
Dest Output
Control ControlB
Length 9
Position 0
EN
DN
Frekvencia otacok motora a lediek Timer
RES
MOV
Move
Source Output 0
Dest Local:2:O.Data 2#0000_0000_0000_0000
6

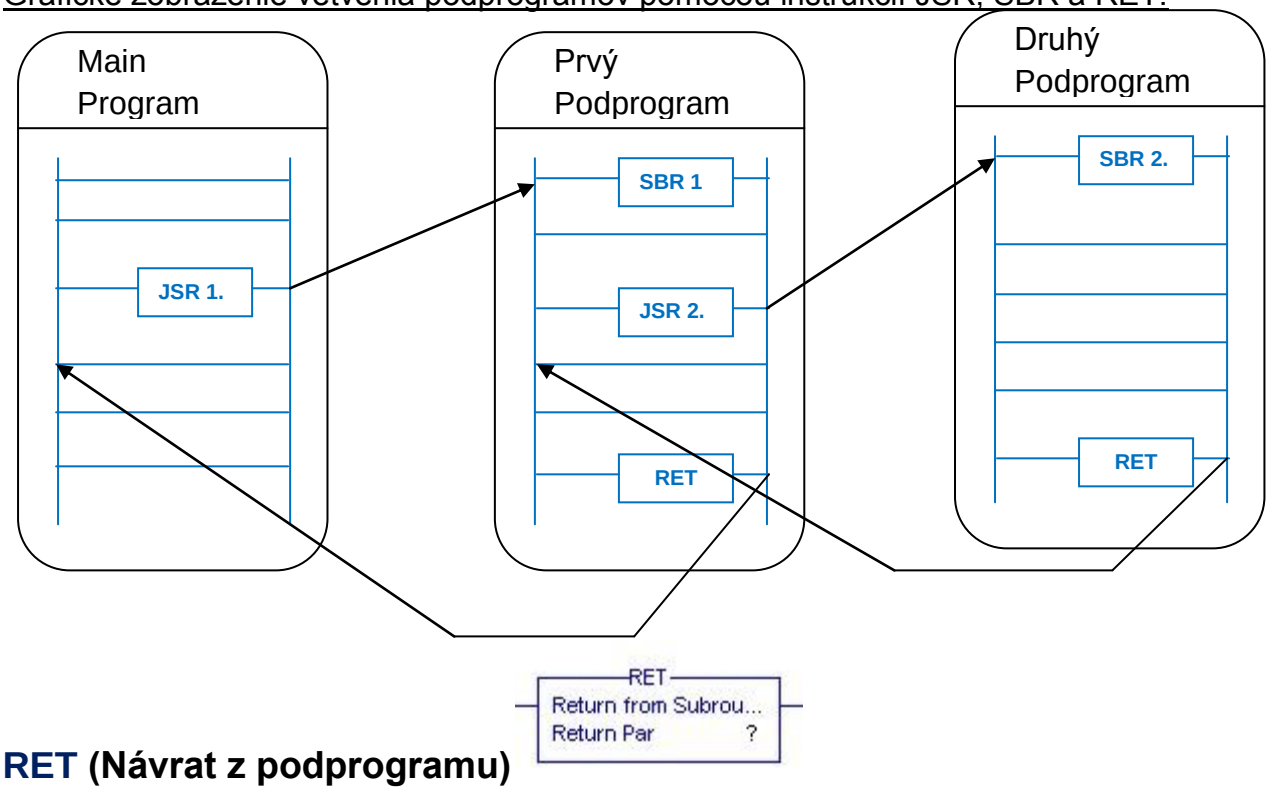

Vynulovanie sekvencie SQO pre opačný smer otačania
Smer motora
EQU
Equal
Source A Smer 0
Source B 1
Obrateny smer otacania
EQU
Equal
Source A ControlB.POS 0
Source B 9
Obrateny smer otacania
MOV
Move
Source 0
Dest ControlB.POS 0
7
Po stačení tlačidla TL4 sa ukončí program
Spustenie programu Start
Tlacidlo pre vysledky alebo ukoncenie programu TL4
<Local:1:I.Data.3>
MOV
Move
Source 0
Dest Local:2:O.Data 2#0000_0000_0000_0000
Spustenie programu Start
U
8
Návrat z podprogramu
Programova sada inštrukcií sekvencií SS_Prog[1]
Spustenie programu Start
U
RET
Return from Subrou...
9
(End)

1. Inštrukcie riadenia programu

JSR (Programový skok do podprogramu)

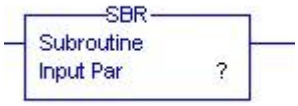
JSR je inštrukcia, ktorá sa vykoná, ak stav rungu je priechodný. Táto inštrukcia zabezpečuje programový skok do podprogramu. Inštrukcia sa používa spolu z inštrukciami SBR a RET. Touto inštrukciou skáčeme na prvý rung a prvú inštrukciu v zadanom podprograme. Do slova Routine Name sa ukladá dátový typ ROUTINE, tzn. názov podprogramu, do ktorého chceme skočiť. Taktiež môžeme odosielať do podprogramu potrebné vstupné parametre, hodnoty z vykonávajúceho sa programu, ktoré sa musia uložiť do slov Input Par, ktorých môžeme mať ľubovoľný počet. Nasledovné v podprograme potrebujeme na získanie týchto parametrov použiť inštrukciu SBR. Taktiež môžeme získavať výstupne parametre z podprogramu pomocou slov Return Par, ktorých počet je taktiež ľubovoľný. Potom je nevyhnutné použitie RET inštrukcie v podprograme na odoslanie potrebných parametrov, ktoré chceme získať z podprogramu.

Grafické zobrazenie vetvenia podprogramov pomocou inštrukcií JSR, SBR a RET:



RET (Návrat z podprogramu)

RET je inštrukcia, ktorá umožňuje odoslať parametre a premenné z podprogramu do nadradeného programu. Použitie tejto inštrukcie je nevyhnutné pre správne ukončenie podprogramu a návrat do nadradeného programu.



SBR (Podprogram)

SBR je inštrukcia, ktorá umožňuje získať parametre, premenné z nadradeného programu do podprogramu, pre nasledovné používanie. Používanie tejto inštrukcie nie je nutné, ak ne nemáme žiadne vstupné parametre do podprogramu.

Posuvne inštrukcie poľa (súboru)

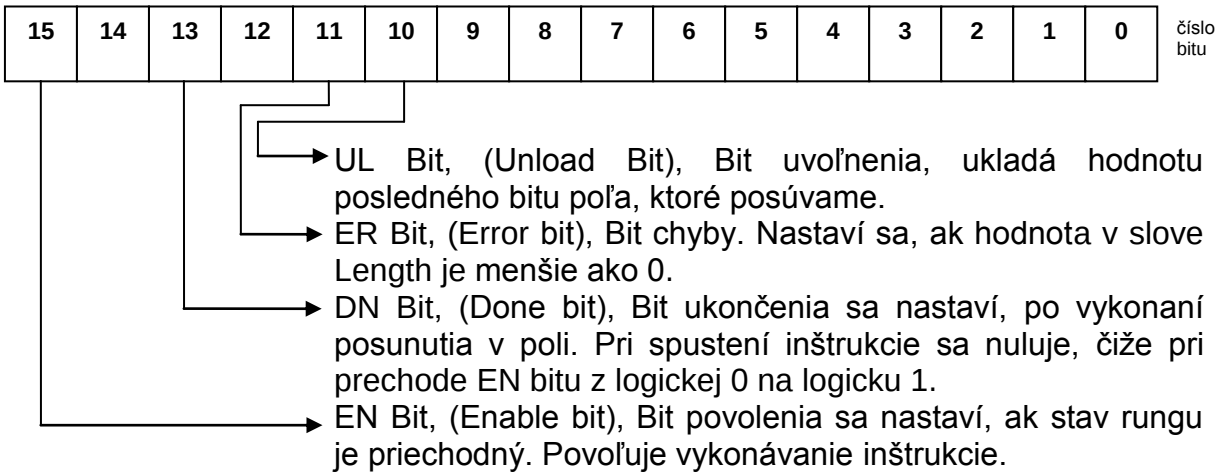
BSL (Bitový posun do láva)

BSL je výstupná inštrukcia, ktorá pri stavu rungu z nepriechodného na priechodný, vykoná posun bitu v poli do ľavej strany, čo znamená že, posledný bit poľa uloží do UL bitu, ostatne bity posunie o jeden register, prvok poľa do ľavej strany a na nultú pozíciu načíta hodnotu adresovanú v Source Bit slove . V Array slove sa adresuje pole v ktorom posúvame jednotlivé bity. Control slovo adresuje ovládaciu štruktúru. V Source Bit slove je adresovaný nový vstupný bit, ktorý sa ukladá po posunutí poľa na nultú pozíciu. Length slovo adresuje počet bitov v poli, ktorý posúvame. Inštrukcie BSL pracuje na súvislé pamäti. Ak posúvane pole je členom väčšieho, nadradeného poľa, ako napríklad pole v ovládacej štruktúre, je potom možné, že inštrukcia BSL môže posunúť bity za hranice posúvaného poľa do nasledujúceho poľa, ktoré patri nadradenému poľu. Pre zabránenie tejto situácie, treba dôkladne nastaviť dĺžku, hodnotu adresovanú slovom Lenght.

Ovládacia štruktúra:

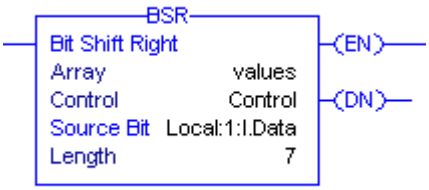
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	číslo bitu
					—	N	N	N	N	N	N	N	N	N	N	N	
Slovo 0	**EN**	**-**	**DN**	**-**	**ER**	**UL**	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	
Slovo 1	**Veľkosť bitového poľa (Length)**																
Slovo 2	**Rezerva (Reserved)**																

Sekvenčne slovo 0:



Sekvenčné slovo 1:

Lenght udáva počet prvkov v bitovom poli.



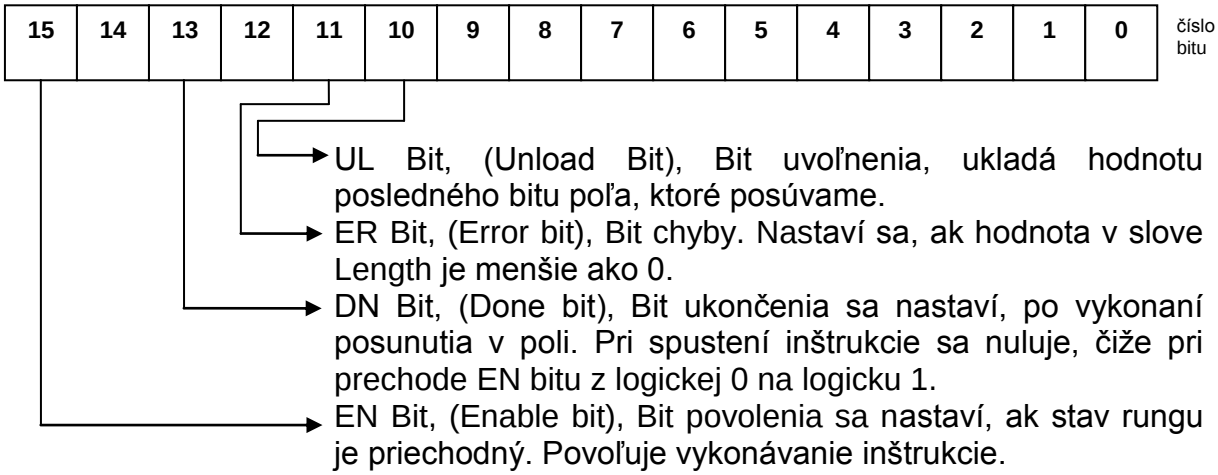
BSR (Bitový posun do prava)

BSR je výstupná inštrukcia, ktorá pri stavu rungu z nepriechodného na priechodný, vykoná posun bitu v poli do pravejj strany, čo znamená že, prvý bit poľa uloží do UL bitu, ostatne bity posunie o jeden register, prvok poľa do pravej strany a na poslednú pozíciu načíta hodnotu adresovanú v Source Bit slove . V Array slove sa adresuje pole v ktorom posúvame jednotlivé bity. Control slovo adresuje ovládaciu štruktúru. V Source Bit slove je adresovaný vstupný bit, ktorý sa ukladá po posunutí poľa na poslednú pozíciu poľa. Length slovo adresuje počet bitov v poli, ktorý posúvame. Inštrukcie BSR pracuje na súvislé pamäti. Taktiež treba dôkladne nastaviť dĺžku, hodnotu adresovanú slovom Lenght, aby sa neposúvali bity ktoré nechceme.

Ovládacia štruktúra:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	číslo bitu
	—				—	N	N	N	N	N	N	N	N	N	N	N	
Slovo 0	**EN**	**-**	**DN**	**-**	**ER**	**UL** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	**-** N	
Slovo 1	**Veľkosť bitového poľa (Length)**																
Slovo 2	**Rezerva (Reserved)**																

Sekvenčne slovo 0:



Sekvenčné slovo 1:

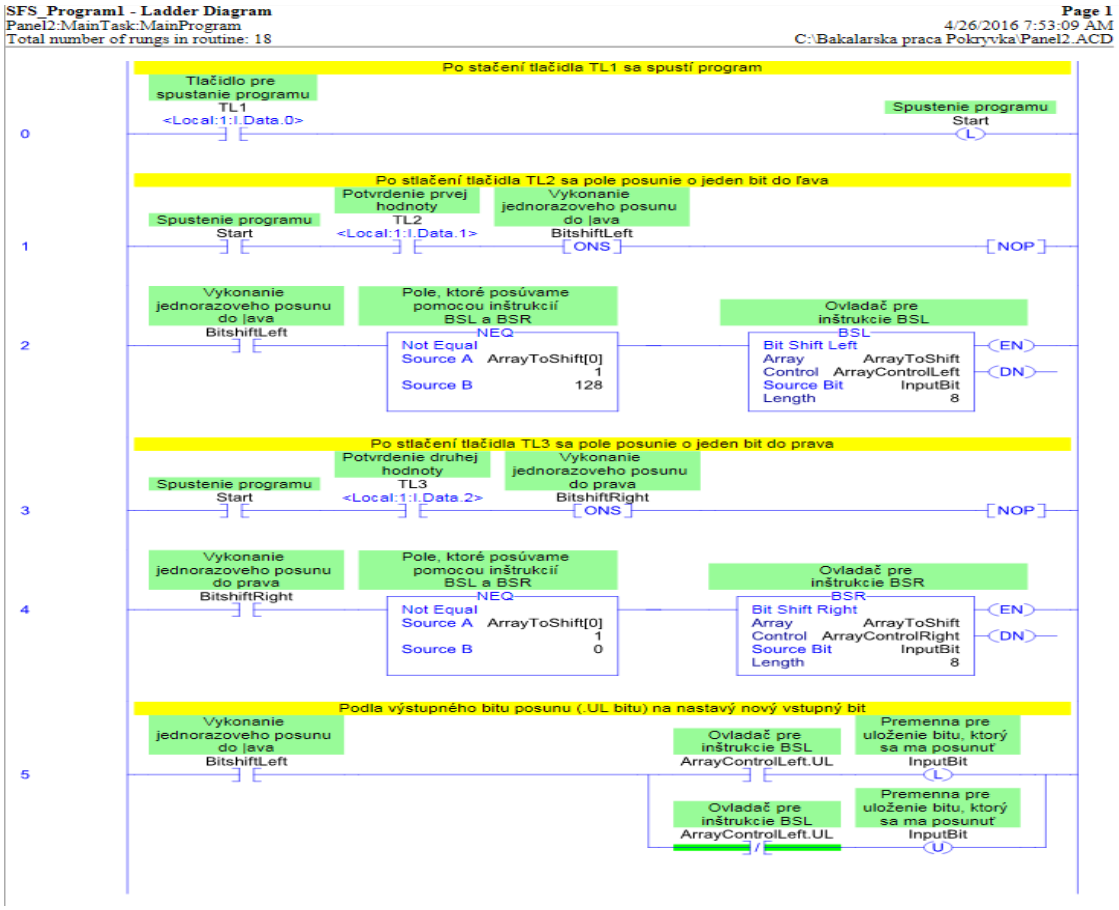
Lenght udáva počet prvkov v bitovom poli.

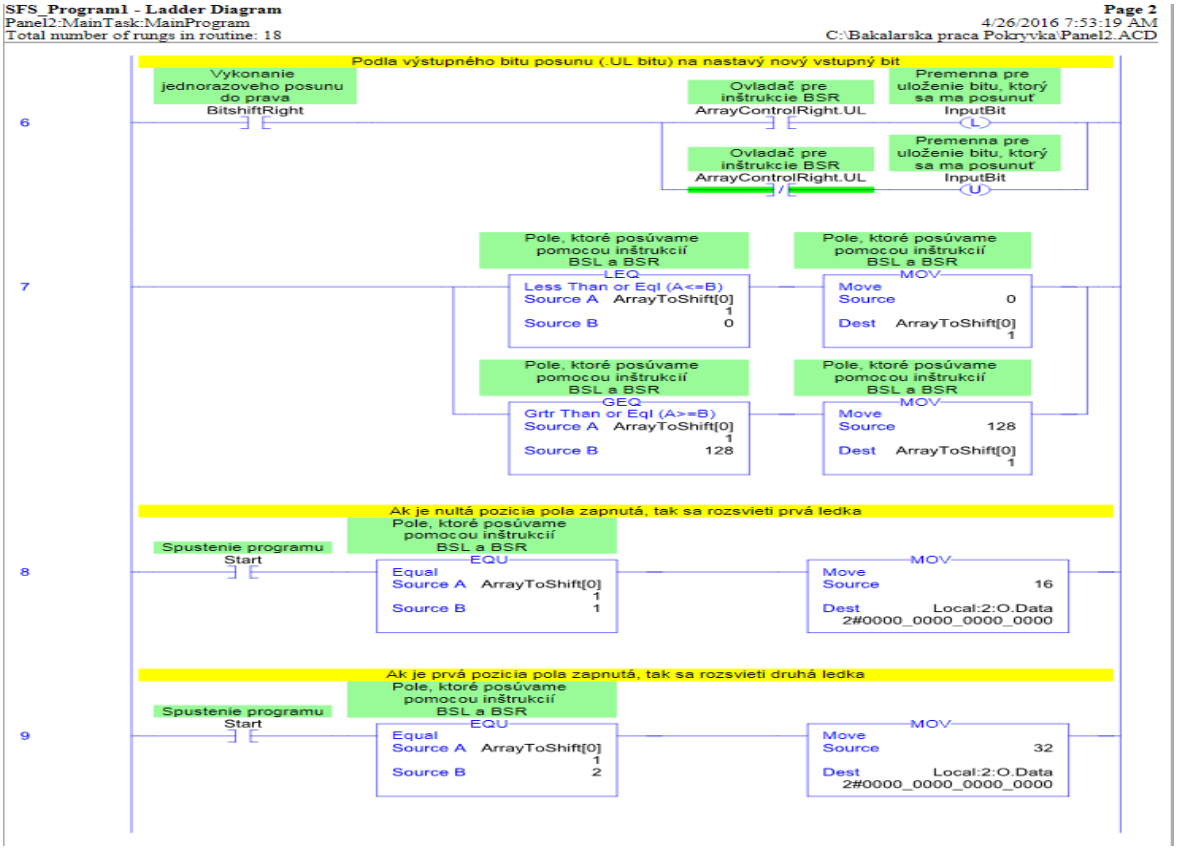
SFS_Program1

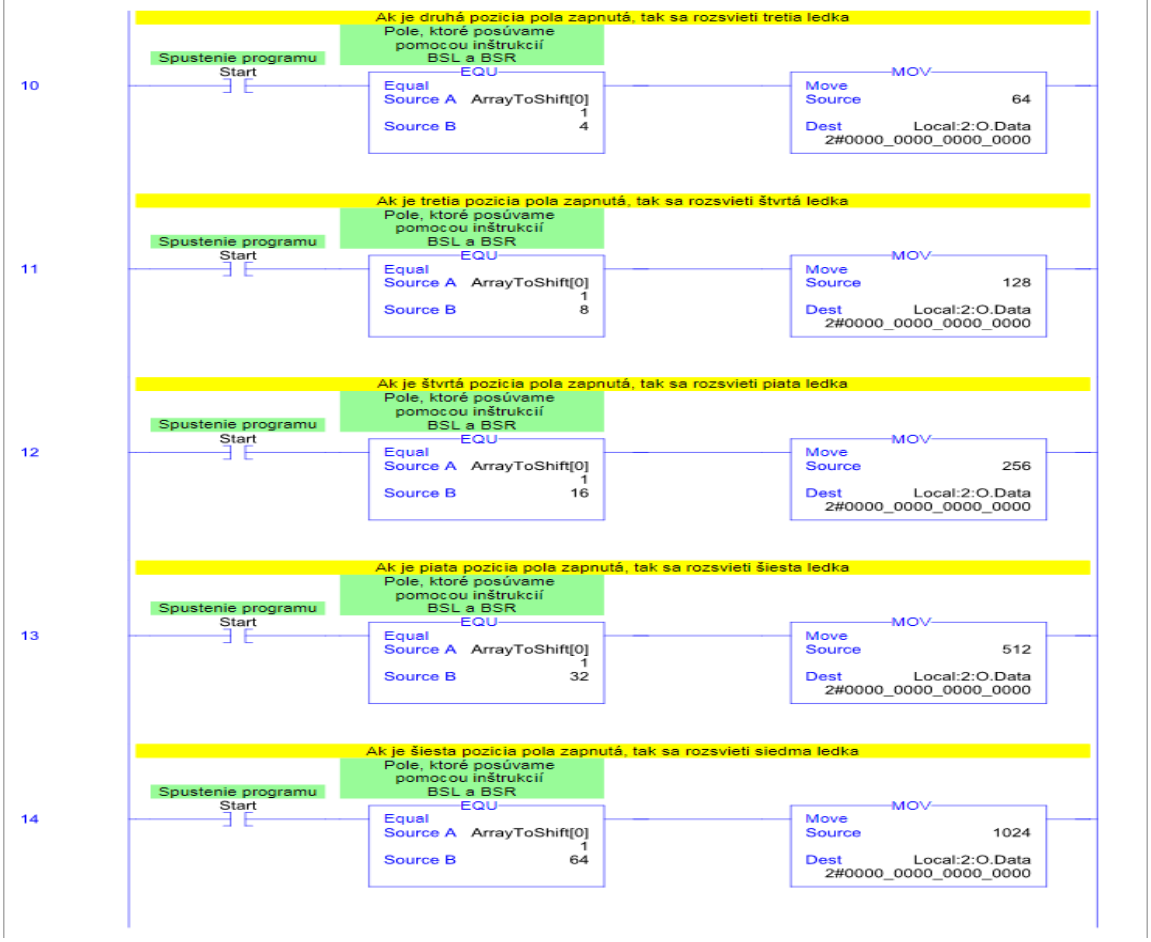
Úloha: Úlohou programu je znázornenie funkcionality inštrukcií BSL a BSR, ktoré posúvajú bity zadaného poľa.

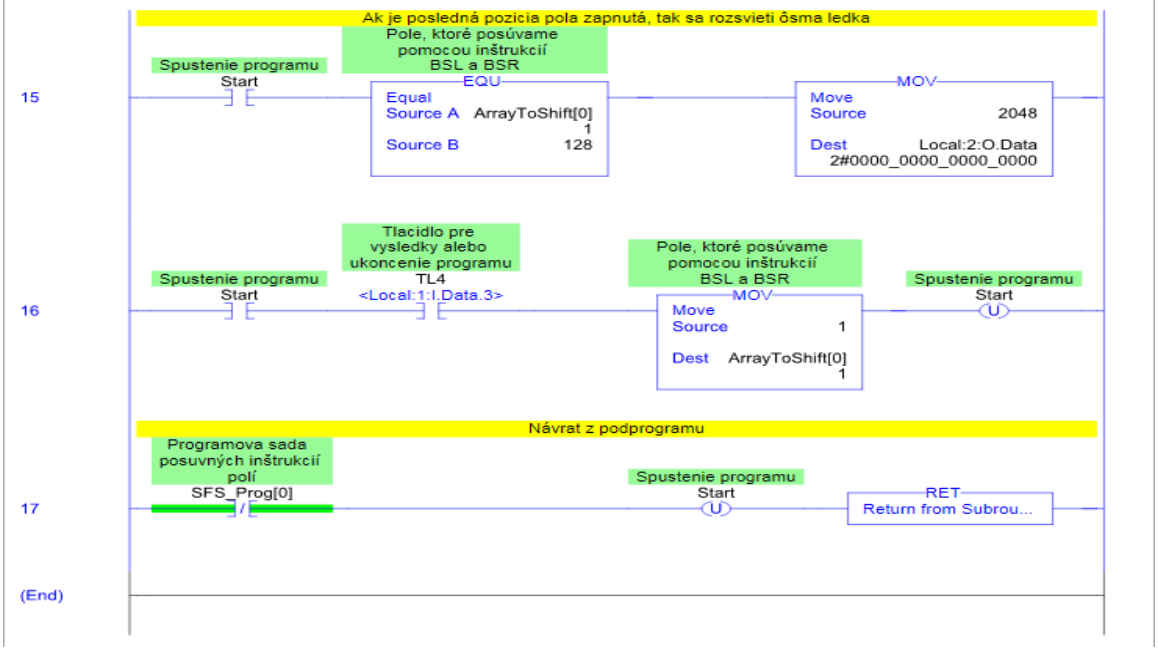
Riešenie: Úlohu bude realizovaná na príklade led hada na panely č.2. Pomocou tlačidla TL1 sa spustí program. Po spustení programu sa bude čakať na zadanie posunu poľa ArrayToShift[0], buď do pravá alebo do ľavá, pričom posun bitov poľa do ľavá je realizovaný stlačením tlačidla TL2 a posun bitov poľa do pravá je realizovaný stlačením tlačidla TL3. Podľa bitu, ktorý vystúpi z pola sa nastaví vstupný bit uložený v premennej InputBit. Po stlačení tlačidla TL4 sa program zastaví. Do podprogramu SFS_Program1 vstúpime, ak premenná SFS_Prog[0] nadobúda logickú jednotku.

Rebríková schéma podprogramu SFS_Program1:






Ak je druhá pozicia pola zapnutá, tak sa rozsvieti tretia ledka
Pole, ktoré posúvame pomocou inštrukcií BSL a BSR
Spustenie programu
Start
10
EQU
Equal
Source A ArrayToShift[0] 1
Source B 4
MOV
Move
Source 64
Dest Local:2:O.Data 2#0000_0000_0000_0000
Ak je tretia pozicia pola zapnutá, tak sa rozsvieti štvrtá ledka
11
Source B 8
Source 128
Ak je štvrtá pozicia pola zapnutá, tak sa rozsvieti piata ledka
12
Source B 16
Source 256
Ak je piata pozicia pola zapnutá, tak sa rozsvieti šiesta ledka
13
Source B 32
Source 512
Ak je šiesta pozicia pola zapnutá, tak sa rozsvieti siedma ledka
14
Source B 64
Source 1024


Ak je posledná pozicia pola zapnutá, tak sa rozsvieti ôsma ledka
Pole, ktoré posúvame pomocou inštrukcií BSL a BSR
Spustenie programu
Start
15
EQU
Equal
Source A ArrayToShift[0] 1
Source B 128
MOV
Move
Source 2048
Dest Local:2:O.Data 2#0000_0000_0000_0000
Tlacidlo pre vysledky alebo ukoncenie programu
TL4
<Local:1:I.Data.3>
16
Source 1
Dest ArrayToShift[0] 1
Návrat z podprogramu
Programova sada posuvných inštrukcií polí
SFS_Prog[0]
17
RET
Return from Subrou...
(End)