```
FUNCTION_BLOCK "FB_Data_Shift"
TITLE =
VERSION : 0.1


VAR_INPUT
  i_DAT_TYP : INT ;
  i_DAT_LEN : INT ;
  i_DB_NO : INT ;
  i_DB_BYTE : INT ;
  i_DB_BIT : INT ;
  i_DAT_LEN_TR : INT ;
  i_IN_V : INT ;
  i_EN : BOOL ;
END_VAR
VAR_TEMP
  T_AR1 : DWORD ;
  T_AR2 : DWORD ;
  t_OUT_ANY : ANY ;
  t_BLOCK_DB_NO : WORD ;
  t_LOOP : INT ;
  t_LOOP_LIM : INT ;
  t_DAT_LEN_SUB_2 : INT ;
  t_DB_BYTE_LOW : DWORD ;
  t_DB_BYTE_LEN : DWORD ;
  t_ADDRESS_SOURCE : DWORD ;
  t_ADDRESS_DESTINATION : DWORD ;
  t_Base_Address : DWORD ;
  t_DAT_LEN : INT ;
  t_DB_NO : INT ;
  t_DAT_TYP : INT ;
  t_DB_BYTE : INT ;
  t_DB_BIT : INT ;
END_VAR
BEGIN
NETWORK
TITLE =

      TAR1  #T_AR1; // SAVE ADRESSREGISTER AR1
      TAR2  #T_AR2; // SAVE ADRESSREGISTER AR2
NETWORK
TITLE =

      L     #i_DAT_LEN;
      T     #t_DAT_LEN;

      L     #i_DB_NO;
      T     #t_DB_NO;

      L     #i_DAT_TYP;
      T     #t_DAT_TYP;

      L     #i_DB_BYTE;
      T     #t_DB_BYTE;

      L     #i_DB_BIT;
      T     #t_DB_BIT;
```

```
NETWORK
TITLE =

      L     P##t_OUT_ANY; //  START ADRESS OF ANY-POINTER
      LAR2  ; // INTO AR2

      L     B#16#10; // ENTER SYNTAX-ID
      T     B [AR2,P#0.0]; // INTO ANY-POINTE 0 BYTES

      L     #t_DAT_TYP; //Input data type:
      1=bool；2=byte；3=CHAR；4=Word;5=Int;6=DWORD;7=DINT;8= Real;9=Date
      L     W#16#0;
      <>I   ;
      JC    NIL;
      L     B#16#0;
      JU    TYPE;
NIL:  L     #t_DAT_TYP; //If data type is Bool
      L     W#16#1;
      <>I   ;
      JC    BOOL;
      L     B#16#1;
      JU    TYPE;
BOOL: L     #t_DAT_TYP; //If data type is Byte
      L     W#16#2;
      <>I   ;
      JC    BYTE;
      L     B#16#2;
      JU    TYPE;
BYTE: L     #t_DAT_TYP; //If data type is Char
      L     W#16#3;
      <>I   ;
      JC    CHAR;
      L     B#16#3;
      JU    TYPE; //If data type is W
CHAR: L     #t_DAT_TYP;
      L     W#16#4;
      <>I   ;
      JC    WORD;
      L     B#16#4;
      JU    TYPE;
WORD: L     #t_DAT_TYP; //If data type is INT
      L     W#16#5;
      <>I   ;
      JC    INT;
      L     B#16#5;
      JU    TYPE;
INT:  L     #t_DAT_TYP; //If data type is Word
      L     W#16#6;
      <>I   ;
      JC    DWOR;
      L     B#16#6;
      JU    TYPE;
DWOR: L     #t_DAT_TYP; //If data type is DINT
      L     W#16#7;
      <>I   ;
```

```
      JC    DINT;
      L     B#16#7;
      JU    TYPE;
DINT: L     #t_DAT_TYP; //If data type is real
      L     W#16#8;
      <>I   ;
      JC    REAL;
      L     B#16#8;
      JU    TYPE;
REAL: L     #t_DAT_TYP; //If data type is date
      L     W#16#9;
      <>I   ;
      JC    DATE;
      L     B#16#9; // ELSE LOAD AREA TYPE "DATE"
      JU    TYPE; // USE TYPE
DATE: L     #t_DAT_TYP; // LOAD TYPE OF DATA
      L     W#16#A; // IF TYPE OF DATA
      <>I   ; // <> "TIME_OF_DAY" THEN
      JC    TOD; // ASK FOR TOD (TIME_OF_DAY)
      L     B#16#A; // ELSE LOAD AREA TYPE "TIME_OF_DAY"
      JU    TYPE; // USE TYPE
TOD:  L     #t_DAT_TYP; // LOAD TYPE OF DATA
      L     W#16#B; // IF TYPE OF DATA
      <>I   ; // <> "TIME" THEN
      JC    TIME; // ASK FOR TIME
      L     B#16#B; // ELSE LOAD AREA TYPE "TIME"
      JU    TYPE; // USE TYPE
TIME: L     #t_DAT_TYP; // LOAD TYPE OF DATA
      L     W#16#C; // IF TYPE OF DATA
      <>I   ; // <> "S5TIME" THEN
      JC    S5TI; // ASK FOR S5TIME
      L     B#16#C; // ELSE LOAD AREA TYPE "S5TIME"
      JU    TYPE; // USE TYPE
S5TI: L     #t_DAT_TYP; // LOAD TYPE OF DATA
      L     W#16#E; // IF TYPE OF DATA
      <>I   ; // <> "DATE_AND_TIME" THEN
      JC    DT; // ASK FOR DT (DATE_AND_TIME)
      L     B#16#E; // ELSE LOAD AREA TYPE "DATE_AND_TIME"
      JU    TYPE; // USE TYPE
DT:   L     #t_DAT_TYP; // LOAD TYPE OF DATA
      L     W#16#13; // IF TYPE OF DATA
      <>I   ; // <> "STRING" THEN
      JC    STRI; // ASK FOR STRING
      L     B#16#13; // ELSE LOAD AREA TYPE "STRING"
      JU    TYPE; // USE TYPE
STRI: L     #t_DAT_TYP; // LOAD PARAMETER TYPE OF DATA
      L     W#16#17; // IF PARAMETER TYPE OF DATA
      <>I   ; // <> "BLOCK_FB" THEN
      JC    BLFB; // ASK FOR BLOCK_FB
      L     B#16#17; // ELSE LOAD PARAMETER TYPE "BLOCK_FB"
      JU    TYPE; // USE TYPE
BLFB: L     #t_DAT_TYP; // LOAD PARAMETER TYPE OF DATA
      L     W#16#18; // IF PARAMETER TYPE OF DATA
      <>I   ; // <> "BLOCK_FC" THEN
      JC    BLFC; // ASK FOR BLOCK_FC
      L     B#16#18; // ELSE LOAD PARAMETER TYPE "BLOCK_FC"
      JU    TYPE; // USE TYPE
```

```
BLFC: L     #t_DAT_TYP; // LOAD PARAMETER TYPE OF DATA
      L     W#16#19; // IF PARAMETER TYPE OF DATA
      <>I   ; // <> "BLOCK_DB" THEN
      JC    BLDB; // ASK FOR BLOCK_DB
      L     B#16#19; // ELSE LOAD PARAMETER TYPE "BLOCK_DB"
      JU    TYPE; // USE TYPE
BLDB: L     #t_DAT_TYP; // LOAD PARAMETER TYPE OF DATA
      L     W#16#1A; // IF PARAMETER TYPE OF DATA
      <>I   ; // <> "BLOCK_SDB" THEN
      JC    BSDB; // ASK FOR BLOCK_SDB
      L     B#16#1A; // ELSE LOAD PARAMETER TYPE "BLOCK_SDB"
      JU    TYPE; // USE TYPE
BSDB: L     #t_DAT_TYP; // LOAD PARAMETER TYPE OF DATA
      L     W#16#1C; // IF PARAMETER TYPE OF DATA
      <>I   ; // <> "COUNTER" THEN
      JC    COUN; // ASK FOR COUNTER
      L     B#16#1C; // ELSE LOAD PARAMETER TYPE "COUNTER"
      JU    TYPE; // USE TYPE
COUN: L     #t_DAT_TYP; // LOAD PARAMETER TYPE OF DATA
      L     W#16#1D;
      <>I   ;
      JC    TIMR;
      L     B#16#1D;
      JU    TYPE;
TIMR: L     B#16#2;
TYPE: T     B [AR2,P#1.0]; //After check variable type then load data type to AR2 offset 1.0
      L     #t_DAT_LEN;
      T     W [AR2,P#2.0]; //Load Data length to offset 2.0, data type is WORD
      L     #t_DB_NO;
      T     W [AR2,P#4.0]; //Load data number to offset 4.0 data type is WORD
      T     #t_BLOCK_DB_NO; //Also tranfer to #t_BLOCK_DB_NO
      L     0;
      ==I   ; //If Data block is 0, then jump to FLAD
      JC    FLAD;
      L     P#DBX 0.0; //If Data block is not zero Load DBX0.0 address
      JU    OFFD;
FLAD: L     P#M 0.0;
OFFD: L     #t_DB_BYTE; //Load i_DB_TYPE is INT
      SLD   3; //Change to address: such as i_DB_TYPE=2, after change value is 2.0
      +D    ;
      L     #t_DB_BIT; //Add bit address
      +D    ;
      T     D [AR2,P#6.0]; //Transfer to AR2 offset 6.0, data type is DWORD
NETWORK
TITLE =


      OPN   DB [#t_BLOCK_DB_NO];
NETWORK
TITLE =

      L     D [AR2,P#6.0];
      LAR1  ; //Load Network2's result the address to AR1.
      TAR1  #t_ADDRESS_DESTINATION; //Load AR1 value store to #t_ADDRESS_DESTINATION

      LAR2  #T_AR2; // RESTORE ADRESSREGISTER ADR2
```

```
      L     #i_DAT_LEN;
      L     2;
      -I    ;
      T     #t_DAT_LEN_SUB_2; //Normal, Siemense data is base on Word priciple.

      L     #t_DAT_LEN_SUB_2;
      L     2;
      /I    ;
      T     #t_LOOP_LIM; //Caculate Loop value for move data.here is 7;

      L     #t_DAT_LEN_SUB_2; // #t_DAT_LEN_SUB_2 = 7
      SLD   3;
      AD    DW#16#FFFFFFF;
      T     #t_DB_BYTE_LEN; //Get data length address value, such as data length is 7
      resule is 7.0

      L     #t_DB_BYTE_LEN;
      L     #t_ADDRESS_DESTINATION;
      +D    ;
      T     #t_ADDRESS_DESTINATION; //Load #t_ADDRESS_DESTINATION+16.0

      L     #i_DAT_LEN_TR; //Change   #i_DAT_LEN_TR to address: here #i_DAT_LEN_TR=2, then
      #t_DB_BYTE_LOW=2.0
      SLD   3;
      AD    DW#16#FFFFFF;
      T     #t_DB_BYTE_LOW;

      A     #i_EN; //If allow to tranfer
      JCN   END;

      L     0;
CYCL: T     #t_LOOP; //Initial value is 0
      L     #t_LOOP; //If t_loop value is equal or more than #t_DAT_LEN_SUB_2 then jump to
      A7d0
      L     #t_DAT_LEN_SUB_2;
      >=I   ;
      JC    A7d0;

      L     #t_ADDRESS_DESTINATION; //Load #t_ADDRESS_DESTINATION=#t_ADDRESS_DESTINATION+16.0
      L     #t_DB_BYTE_LOW; //#t_DB_BYTE_LOW = 2.0
      -D    ;
      T     #t_ADDRESS_SOURCE; //#t_ADDRESS_SOURCE = #t_ADDRESS_DESTINATION+14.0

      L     DBD [#t_ADDRESS_SOURCE];
      T     DBD [#t_ADDRESS_DESTINATION]; //Load the source data to destination address

      L     #t_ADDRESS_SOURCE; //Load the source data address to destination address
      T     #t_ADDRESS_DESTINATION;

      L     #t_LOOP; //Loop = Loop - 1
      L     1;
      +I    ;
      JU    CYCL;
A7d0: L     #i_IN_V; //After loop then Load  #i_IN_V to Last data.
      T     W [AR1,P#0.0];
END:  NOP   0;
```

```
NETWORK
TITLE =

      LAR1  #T_AR1; // RESTORE ADRESSREGISTER ADR1



END_FUNCTION_BLOCK
```