```
type
    TCP_Buffer : array [1..522] of byte;
end_type

var_output
    Send_Buffer:        TCP_Buffer  (* Buffer to be handed over to FB_INIT_SEND_RECEIVE *)
    Data_ok:            BOOL        (* Flag indicating data integrity *)
end_var

var_input
    SID:                BYTE        (* ARCNET node ID of sender *)
    DID:                BYTE        (* ARCNET node ID of recipient (destination) *)
    Data_Len:           WORD        (* Size of payload data in bytes *)
    Data_Array:         TCP_Buffer  (* Byte array containing payload data *)
end_var

var
    I:                  WORD        (* internal counter *)
    K:                  WORD        (* internal counter *)
end_var


(* check size of Data_Array according to ARCNET spec. *)

if(Data_Len > word#0 and Data_Len < word#508 and not (Data_Len > word#252 and Data_Len <
word#256)) then

    Send_Buffer[1]  :=  word_to_byte ((WORD#10 + Data_Len) and word#255);  (* IP packet
    length, lower Byte*)
    Send_Buffer[2]  :=  word_to_byte ((WORD#10 + Data_Len) /   word#256);  (* IP packet
    length, upper Byte*)

    Send_Buffer[3]  :=  BYTE#83;    (* 'S' *)
    Send_Buffer[4]  :=  BYTE#5;     (* OP_WRITE_DATA *)

    Send_Buffer[5]  :=  word_to_byte ((WORD#6 + Data_Len) and word#255);   (* OpCode packet
    length, lower Byte*)
    Send_Buffer[6]  :=  word_to_byte ((WORD#6 + Data_Len) /   word#256);   (* OpCode packet
    length, upper Byte*)

    Send_Buffer[7]  :=  SID;    (* SID *)
    Send_Buffer[8]  :=  DID;    (* DID *)

    Send_Buffer[9]  :=  BYTE#0; (* reserved *)
    Send_Buffer[10] :=  BYTE#0; (* reserved *)
    Send_Buffer[11] :=  BYTE#0; (* reserved *)
    Send_Buffer[12] :=  BYTE#0; (* reserved *)


    for I := word#1 to Data_Len by word#1 do (* take-over of Data_Array into Send_Buffer *)

        K := I+word#12;
        Send_Buffer[K] := Data_Array[I];

    end_for;
```

```
    Data_ok := true;

else
    Data_ok := false; (* Length of payload data not according to ARCNET spec. *)
end_if;
```