AS511 Protocol Notes
By Luca Gallina
http://www.runmode.com
Last revised: March 17, 2004

Credits
To me, as I first released AS511 doc on the web ☺
To Cesar Garcia for adding details

AS511 hardware
The physical layer relies on a 20mA current loop interface, a flow of 20mA means a logical “0” while the absence of current flow means a logical “1”.

Current loop interface allows longer cables than RS232, at 9600 baud it is rated up to 1000 meters.

Siemens programming devices (PG) have a built-in Current Loop port (old PG 675, 685, 730, 750 etc actually have a non-standard 25 pin connector suitable for both 20mA and RS232 interfaces).

You may buy a RS232/20mA C.L. converter from Siemens or from third parties manufacturer (e.g. Patton) or you can make your own converter, cheap but fully functional.
I built and used homebuilt converters for years, some schematics are available at http://www.runmode.com, but be aware that I give no support, obligation or liability. Other converters schematics can be easily found on the web. Build and use them at your own risk.

Passive converters can be used to communicate with S5 90, 95, 100, 115 families CPUs
Active converters are needed to communicate with S5 135, 150, 155 families CPUs and all intelligent modules such are IP, CP and WF cards.
OP panels will need active converters as well.

AS511 serial data frame
AS511 uses an uncommon 11 bit serial frame:

Speed	9600 baud
Data	8 bit
Stop bits	1
Parity	Even

Not a problem with current hardware, but in late ‘80s and early ’90 not all PC serial cards supported 11 bit frames.

AS511 protocol
AS511 is the programming port protocol for all S5 PLC’s.
A lot of overhead time is spent in opening and terminating each transmission, one could say that AS511 partners spend most of the time in handshaking rather than transferring actual data.

The DLE character
An important part is played by <DLE> (10 Hex), a control character used to inform the receiver that the next incoming character is to be considered a protocol's control character and not a data byte. In other words, <DLE> is a software switch that AS511 protocol uses to separate control characters from data bytes.

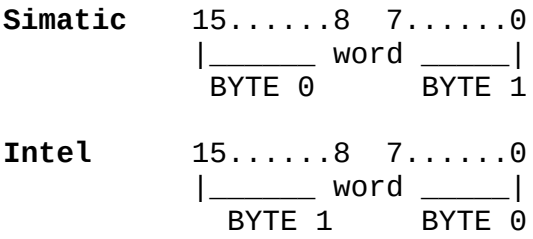
Sender
Whenever a data byte containing the value 10hex is to be sent, the sender must double the char. This means that to send a data byte with value 10Hex the sender must actually send two bytes 10H 10H (<DLE> <DLE>).

Receiver
The receiver is aware that a single <DLE> will mean a protocol signal, while two consecutive <DLE> <DLE> must be accepted as one data byte with 10H value. This is an hard work for the receiver, since it never knows exactly how many bytes to expect and all incoming bytes must be analysed before being considered data values.

Take also note that the PLC can answer <DLE> <NAK> instead of <DLE> <ACK> if the requested operation can not be carried out. In this case, the partner must abort the communication and wait for 500ms before attempting a new communication. The value of 500mS is also applied as timeout in case of missing response from the partner.

Data representation
Be aware that Simatic systems use the Big Endian storage scheme (common in Motorola 680x0 processors), while PC and most of PLCs use the Little Endian coding (common in Intel 80x86 processors).

The 680x0 family stores the high byte first while the 80x86 family stores the low byte first, so the bytes order is reversed:



To obtain the correct value of a data word, a PC application must swap the bytes before reading its content. Nonetheless, PC application bytes must be swapped before being sent to the PLC

Memory data allocation
To read or write a datablock value, you need to know its address in PLC memory. Therefore you must invoke B_INFO function to know memory areas address, then you can call DB_READ or DB_WRITE providing the initial and final address of the area to be read or written

PG and AG
In the following description, as stated in Siemens manuals, the programming unit (or the PC) used to access the PLC is named **PG**, while the PLC is named **AG**

function BLOCK INFORMATION (B_INFO)

- input: block number
- output: initial absolute address of the block in PLC memory. In case of datablocks, the function returns the address of the first dataword (DW 0)

If you want to read or write a block (most likely a data block) in the PLC you must call B_INFO at least once in order to know the block absolute address.
Blocks allocation in PLC memory is dynamic , each time a block is downloaded to the PLC it is allocated in a different place in PLC memory. After a memory compression, the memory is compacted and all blocks address are changed accordingly. Therefore B_INFO must be called again after these events to ensure proper addresses validity.

TIP
different codes for the block identification, the first byte in the header section, as follows:

- *ID = 1 Data block (DB)*
- *ID = 2 Structured Block (SB)*
- *ID = 4 Program block (PB)*
- *ID = 8 Function block (FB)*
- *ID = 48 Organisation Block (OB)*
- *ID = 76 Extended Function Block (FX)*
- *ID= 144 Extended Data Blocks (DX)*

```
PG     PLC
(hex) (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
1A ----> B_INFO function code = 1Ah
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------------- header info --------
01 ----> ID=01h for Datablock
XX ----> DB number (0..255)
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
---------------------- data --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 00 NUL
```

```
<---- XX DB initial address
<---- XX
<---- 70 synchronization code: 70 70
<---- 70
<---- 41 block ID and DB number : 41 XX
<---- XX
<---- XX PG ID code: XX XX
<---- XX
<---- XX library number: XX XX
<---- XX
<---- XX block length (words): XX XX
<---- XX
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function DATABLOCK READ (DB_READ)

- input: initial absolute address in PLC memory , final absolute address in PLC memory
- output: contents of datawords

TIP   *DB_READ function can also be used to read data from any memory position in the PLC, just provide the memory address you want to read. This mean you can read Inputs and Outputs areas as well.*

```
 PG      AG
(hex)  (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
04 ----> DB_READ function code = 04h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------------- header info --------
XX ----> initial address in AG: XX XX
XX ---->
XX ----> final address in AG: XX XX
XX ---->
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
---------------------- data --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 00 NUL
<---- 00 NUL
<---- 00 NUL
<---- 00 NUL
<---- 00 NUL
<---- XX first byte of data
<---- XX
<---- XX
. . .
. . .
<---- XX
<---- XX
<---- XX last byte of data
<---- 10 DLE
<---- 03 ETX
```

```
10 ----> DLE
06 ----> ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function DATABLOCK WRITE (DB_WRITE)
input: initial address in AG, contents of datawords
output: none

TIP
DB_WRITE function can also be used to write data in any memory position in the PLC.
This means you must be extremely careful in assigning proper absolute addresses.

```
PG      AG
(hex) (hex)
----------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
03 ----> DB_WRITE function code = 03h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
----------------------- header info --------
XX ----> initial address in AG: XX XX
XX ---->
----------------------- data --------
XX ----> first byte of data
XX ---->
XX ---->
. . .
. . .
XX ---->
XX ---->
XX ----> last byte of data
10 ----> DLE
04 ----> EOT
----> 10 DLE
----> 06 ACK
----------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function SYS_PAR

- input: none
- output: address of I/O, flags, counter and timers.

TIP   *SYS_PAR provides you the address of I/O, flags, counters and timers. You can then use DB_READ and DB_WRITE functions to access to these memory areas.*

```
PG      AG
(hex) (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
18 ----> SYS_PAR function code = 18h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
00 ----> NULL
00 ----> NULL
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
<---- 02 STX
10 ----> DLE
06 ----> ACK
---------------------- data --------
<---- 00 NUL
<---- 00 NUL
<---- 00 NUL
<---- 00 NUL
<---- 00 NUL
<---- XX AdrPAE
<---- XX AdrPAE
<---- XX AdrPAA
<---- XX AdrPAA
<---- XX AdrM
<---- XX AdrM
<---- XX AdrT
<---- XX AdrT
<---- XX AdrZ
<---- XX AdrZ
<---- XX AdrIAData
<---- XX AdrIAData
.....
```

```
..... 26 Don´t care bytes.
.....
<---- XX PLCType
.....
..... 7 Don´t care bytes.
.....
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
----------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function COMPRESS

- input: none
- output: none

```
PG      AG
(hex)  (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
07 ----> COMPRESS function code = 07h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function DELETE_ALL

- input: none
- output: none

```
PG      AG
(hex)  (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
11 ----> DELETE_ALL function code = 11h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h // When the PLC is running the answer is 0D / 02
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function DELETE_BLOCK

- input: Block number and type.
- output: none

```
PG      AG
(hex)  (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
09 ----> DELETE_BLOCK function code = 09h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h // When the PLC is running the answer is 0D / 02
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------------- data --------
ID ----> ID=01h for Datablock
NN ----> Data block numer (1..255)
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function START/STOP PLC

To start and stop PLC there is not a specific funtion in the AS511 protocol. You need the system information using the SYS_PAR function and overwrite the STOZUS, STOANZ and BATBUF bits from the system dataword six (SD 6).
To stop the PLC write (88h,00h) to the address XX0C (SD 6) .
To start the PLC write (48h,00h) to the SD6 using the DB_WRITE function.

Function B_OWRITE

- input: number and type of block.
- output: none

```
PG      AG
(hex) (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
08 ----> B_OWRITE function code = 08h
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------Block information --------
ID ----> ID=01h for Datablock
NN ----> Data block numer (1..255)
LH ----> Data block length (high)
LL ----> Data block length (low)
10 ----> DLE
03 ----> ETX
<---- 10 DLE
<---- 06 ACK
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 09 ???
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
02 ----> STX
<---- 10 DLE
<---- 06 ACK
00 ----> 00 NUL
70 ----> synchronization code: 70 70
70 ---->
01 ----> block ID and DB number : 41 XX
XX ---->
XX ----> PG ID code: XX XX
XX ---->
XX ----> library number: XX XX
XX ---->
XX ----> block length (words): XX XX
XX ---->
DD ----> Start Data
DD ---->
--
```

```
DD ----> End Data
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```

function DIR

- input: type of block.
- Output:

```
PG      AG
(hex)  (hex)
---------------------- function start --------
02 ----> STX
<---- 10 DLE
<---- 06 ACK
1B ----> DIR function code = 1Bh
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 16 AG answer=16h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------Block information --------
ID ----> ID=01h for Datablock
10 ----> DLE
04 ----> EOT
<---- 10 DLE
<---- 06 ACK
<---- 02 STX
10 ----> DLE
06 ----> ACK
---------------------- data --------
<---- 00
<---- 00
<---- XX Adrress of DB1 (high)
<---- XX Adrress of DB1 (low)
<---- XX Adrress of DB2 (high)
<---- XX Adrress of DB2 (low)
. .
<---- XX Adrress of DB255 (high)
<---- XX Adrress of DB255 (low)
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
---------------------- terminate --------
<---- 02 STX
10 ----> DLE
06 ----> ACK
<---- 12 AG "end of transmission" code: 12h
<---- 10 DLE
<---- 03 ETX
10 ----> DLE
06 ----> ACK
```