10.3: Battery Backup SRAM

The I-8417/8817/8437/8837 can integrate with a S256 or S512 battery backup SRAM to store data, alarm, and information, while X607 & X608 for the I-7188EG/XG controller. The data been stored in these SRAM is always retained unless their battery running out of energy. Their memory size is as below, however the upper 12K is reserved by ISaGRAF controllers.

I-8417/8817/8437/8837
- S256: 244K bytes (256-12=244)
- S512: 500K bytes (512-12=500)

I-7188EG/XG
- X607: 116K bytes (128-12=116)
- X608: 500K bytes (512-12=500)

If battery backup SRAM is found in the controller, the maxinum number of retained variables can be extend to as below.

Boolean :	256
Integer + Real :	256
Timer :	32

ICP DAS provides an utility "**ICPDAS UDloader**" that can be installed on the PC to upload and download data from/to the ISaGRAF controller. Please copy "**UDloader.exe**" from the ICP DAS's CD-ROM:\napdos\isagraf\some_utility\ to your windows.

The I-8417/8817/8437/8837 supports S256/S512 since its driver version of 2.25, while I-7188EG supports X607/608 since its driver version of 1.17, and version 1.15 for I-7188XG. If your driver is older one, please upgrade the hardware driver to the associate version or a higher version. The driver can be found from the below ICP DAS's FTP site:

I-8417/8817/8437/8837: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/isagraf/8000/driver/
I-7188EG: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/isagraf/7188eg/driver/
I-7188XG: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/isagraf/7188xg/driver/

The I/O library should be re-installed if yours is older one. Please refer to section 1.2.
Or you can refer to Appendix A.2 to simply install "C functions" with the below items.

S_B_R, S_B_W, S_BY_R , S_BY_W, S_M_R, S_M_W
S_WD_R, S_WD_W, S_N_R , S_N_W, S_R_R, S_R_W
S_DL_EN, S_DL_EN, S_DL_RST, S_DL_STS
S_FL_INI, S_FL_AVL, S_FL_RST, S_FL_STS, S_MV

and "I/O complex equipment" : S256_S512.

10.3.1: Access to the SRAM

The SRAM can store boolean, byte, word, integer, real & message. Their format is as below.

Boolean:	True=1, False=0	1 byte
Byte:	0 ~ 255	1 byte
Word:	-32768 ~ 32767	2 bytes
Integer:	signed 32-bit	4 bytes
Real:	float	4 bytes
Message:	string (len<=255)	len bytes

To access to the SRAM, the below functions can be used (Please refer to Appendix A).

S_B_R, S_B_W, S_BY_R , S_BY_W, S_M_R, S_M_W
S_WD_R, S_WD_W, S_N_R , S_N_W, S_R_R, S_R_W
S_MV

10.3.2: Upload data stored in the SRAM

For PC to upload data stored in the volatile SRAM of the ISaGRAF controllers, the SRAM should be divided into 1 or max to 8 files. Each file has a ID No. of 1 to 8 and a name of max to 12 characters. The below functions are for handling file format inside the SRAM.

S_FL_INI, S_FL_AVL, S_FL_RST, S_FL_STS

Please use functions of S_FL_INI & S_FL_AVL to arrange the file resident location & current available location (Please refer to Appendix A & demo_40, 41 or 42).

The volatile SRAM is consisted of bytes. The total number of bytes available depends on which module is used as below. The upper 12K is reserved.

Module name	Byte No.
I-8xx7: S256	1 ~ 249,856 (244K), (256-244=12K is reserved)
I-8xx7: S512	1 ~ 512,000 (500K), (512-500=12K is reserved)
I-7188XG/EG: X607	1 ~ 118,784 (116K), (128-116=12K is reserved)
I-7188XG/EG: X608	1 ~ 512,000 (500K), (512-500=12K is reserved)

A file can be located at any place inside these bytes. Each file's location can be described as (**Begin**, **End**). Begin is the lower limit byte No. of the associated file, while End is the upper limit byte No., and Begin is always less than End.

A file inside the SRAM has a current available area (**Head**, **Tail**). Head is the starting position of the file, Tail is the ending position. Head can be larger, less than or equal to Tail.

For ex, a file resides at (Begin, End) = (1, 20000)

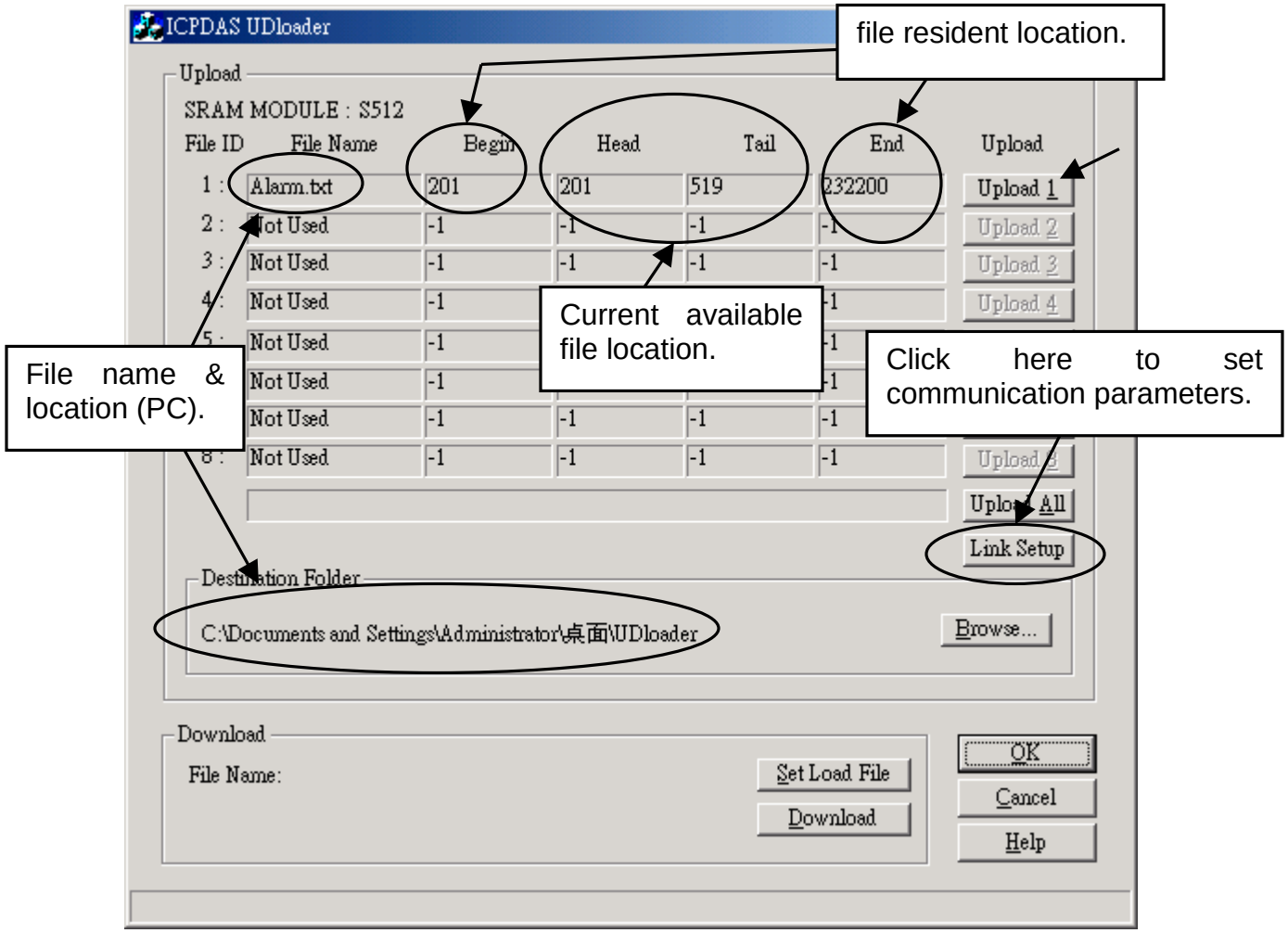
1. If (Head, Tail) = (1001,5100), it means the available data of the file is starting from byte No. of 1001 to 5100. The available file contains 4100 bytes.
2. If (Head, Tail) = (10001,5000), it means the available data of the file is starting from byte No. of 10001 to 20000 and then continued with 1 to 5000. The available file contains 15000 bytes.
3. If (Head, Tail) = (5001,5000), it means the available data of the file is starting from byte No. of 5001 to 20000 and then continued with 1 to 5000. The available file contains 20000 bytes.
4. If (Head, Tail) = (5000,5000), it means the available data of the file is empty, 0 byte.

5. If (Head, Tail) = (-1,-1), it means the available data of the file is empty, 0 byte.

To upload the data stored in the SRAM, please make sure you have installed the “ICPDAS UDloader” on your PC.

To upload data stored in the SRAM of the ISaGRAF controller to PC, please run “UDloader.exe”, then click on “Link Setup” to set proper communication parameters, then click on “Upload 1” to upload it.

Example:
Please download demo_41 to one I-8417/8817/8437/8837. Then push button 1 or 2 or 3 or 4 several times. Then upload the file stored in the SRAM.



10.3.3: Download data to the SRAM

For PC to download data to the volatile SRAM of the ISaGRAF controllers. The below functions can be used. Please refer to Appendix A & demo_44.

S_DL_EN, S_DL_EN, S_DL_RST, S_DL_STS

Please call S_DL_EN to enable it.

The Controller accepts only the binary format for String, Byte, Word, Int & Real.

Byte:	0 ~ 255	1 byte
Word:	-32768 ~ +32767	2 byte [low bye] [high byte]
Int:	32-bit, signed integer	4 byte [lowest] [2nd] [3rd] [highest]
Real:	32-bit float	4 byte [lowest] [2nd] [3rd] [highest]
String:	max to 255 bytes	

If using the "UDloader.exe" to download data to the volatile SRAM, the data to be downloaded should be edited as a text file. Its format should follow the below rules.

1. The first line should be a No. indicate that to download to which starting Byte No. of the SRAM. Valid starting byte No is as below.

S256:	1 ~ 249,856	S512:	1 ~ 512000
X607:	1 ~ 118,784	X608:	1 ~ 512000

2. The other line is the data.
   A. String
      String should start and end with the character of ‘ , for ex. ‘Abcd123’ (7 byte). The $NN (NN in hexidecimal and should not equal to 0), could be used to indicate the ASCII character. For ex, ‘ABC$0D’ contains 4 bytes, the 4th byte is <CR>.
   B. Byte
      Byte should start with (and end with) , for ex. (0) , (123), (255). Valid byte range is from (0) to (255).
   C. Word
      Word should be start with [and end with] , for ex. [-100] , [20000], [32767]. Valid word range is from [-32768] to [32767].
   D. Integer
      Integer should be start with { and end with } , for ex. {-1234567} , {200000}. Valid integer range is from {-2147483648} to {2147483647}.
   E. Real
      Real value should be start with < and end with > , for ex. <123> , <1.56E-2>, <-123.456>.

3. The character between each Byte, Word, Integer, Real, String at the same line should be at least one space character <SP> or , <Comma> or, <Tab>

For ex.

201 ← to download to the SRAM which staring from byte No. 201
‘Hello’ (10) (20) (30) (40) [-10000] {70000} ‘End’ ← data (total 18 bytes)

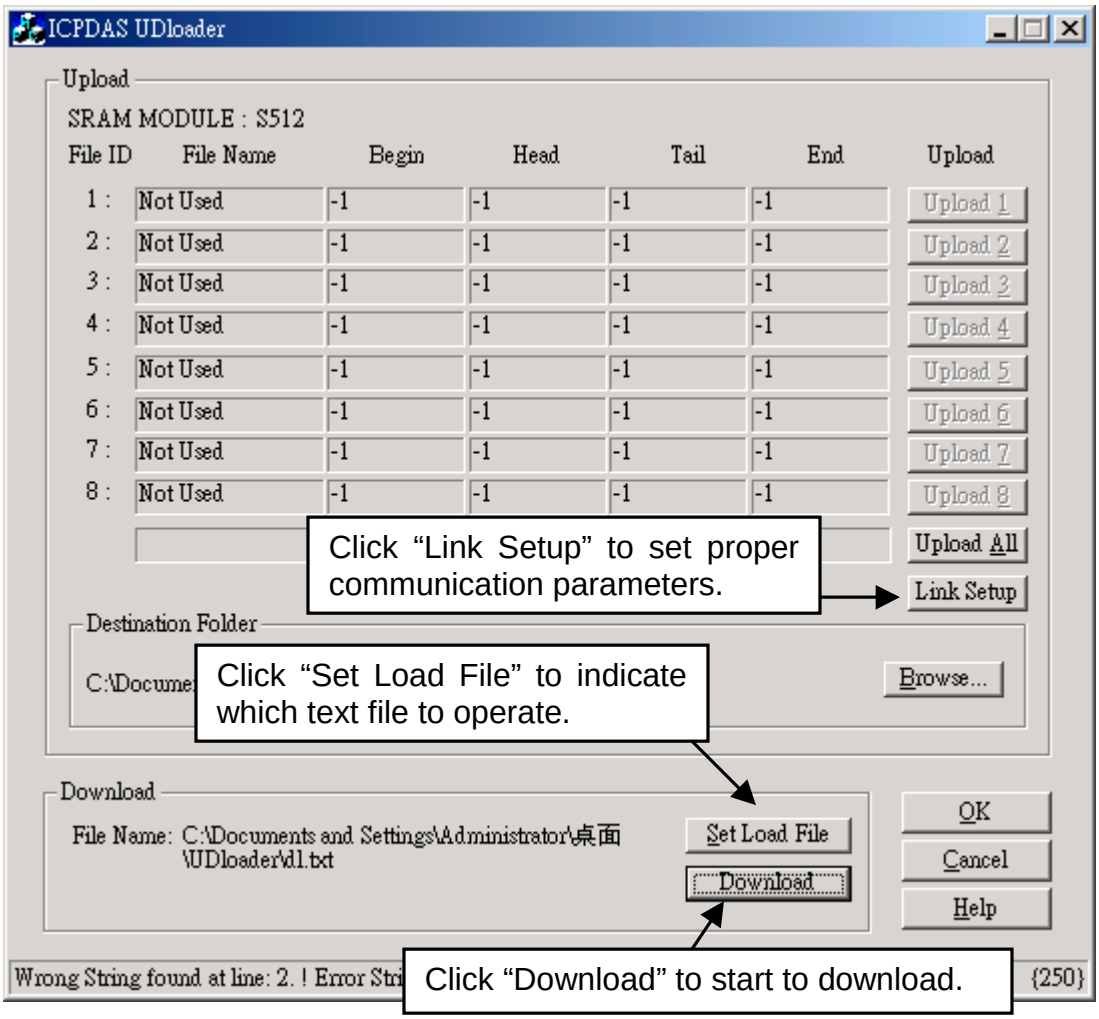
```
1          ← to download to the SRAM which staring from byte No. 1
(10)       ← data (total 57 bytes)
{-1},{2},{-3},{4},{-5},{6} {-7} {8} {-9}   {10}   ← comma, <SP> & <Tab> are all acceptable
<0.123>  <456.789>  <100> , <2.3E3>
```

Example:
Please download demo_44 to one I-8417/8817/8437/8837. Then edit a text file as below.

```
1
{1000}   {250}   {100}   'sTART'
```

The {1000} means the blinking period of L1 is 1000 ms.
The {250} means the blinking period of L2 is 250 ms.
The {100} means the blinking period of L3 is 100 ms. .

Then run "UDloader.exe". You will see something change on the led of the controller.



10.3.4: Operation Functions for the battery backup SRAM

The below functions are for the ISaGRAF controller to access to the volatile SRAM.

S_FL_INI	Init one file's name & location for the volatile SRAM
S_FL_AVL	Set one file's current available byte No. for the volatile SRAM
S_FL_STS	Get file's Status, end byte No. that has been load by PC for the volatile SRAM
S_FL_RST	Reset file's Status to "Not been load by PC yet" for the volatile SRAM

S_B_R:	Read one Boolean (TRUE, FALSE)
S_BY_R:	Read one Byte (0 ~ 255)
S_WD_R:	Read one Word (-32768 ~ +32767)
S_N_R:	Read one Integer (32 bit, signed)
S_R_R:	Read one Real (32 bit, float)
S_M_R:	Read one String

S_B_W:	Write one Boolean (TRUE, FALSE)
S_BY_W:	Write one Byte (0 ~ 255)
S_WD_W:	Write one Word (-32768 ~ +32767)
S_N_W:	Write one Integer (32 bit, signed)
S_R_W:	Write one Real value (32 bit, float)
S_M_W:	Write one String

S_DL_EN	Enable the download permission for PC to download data to the volatile SRAM
S_DL_DIS	Disable the download permission for PC to download data to the volatile SRAM
S_DL_STS	Get PC's Download Status for the volatile SRAM
S_DL_RST	Reset the Download Status to "-1:No action" for the volatile SRAM

S_MV	Move some bytes inside the volatile SRAM

S_FL_INI

Description:
Function **Init one file's name & location for the volatile SRAM**

Arguments:

ID_	**Integer**	File identifier No. (1 ~ 8)
NAME_	**Message**	File name, max to 8 char. for the name & max to 3 char. for the extension. For ex., "data1.txt", "A1234567.bin". Valid char. are A ~ Z , a ~ z , _ , 0 ~ 9, and the 1st should be A ~ Z or a ~ z
BEGIN_	**Integer**	The begin byte No. of the file. BEGIN_ must less than END_
END_	**Integer**	The end byte No. of the file. BEGIN_ must less than END_

S256: 1 ~ 249,856
S512: 1 ~ 512,000
X607: 1 ~ 118,784
X608: 1 ~ 512,000

For ex.,
BEGIN_=101, END_=5000 : the file resides at 101 ~ 5000 inside the SRAM.

return:

Q_	**boolean**	TRUE: ok , FALSE: fail

Example: demo_40, demo_41, demo_42 & demo_44

S_FL_AVL

Description:
Function Set one file's current available byte No. for the volatile SRAM

Arguments:

ID_	**Integer**	File identifier No. (1 ~ 8)
HEAD_	**Integer**	The current available starting byte No.
TAIL_	**Integer**	The current available ending byte No.

(HEAD_, TAIL_) must resides inside the area of the associate file (Please refer to "S_FL_INI"), or Q_ will return FALSE

	1 or
S256:	1 ~ 249,856
S512:	1 ~ 512,000
X607:	1 ~ 118,784
X608:	1 ~ 512,000

For ex.,

A file of ID_ = 1 resides at byte No. of 1 ~ 20000 , it can store max to 20000 bytes.

1. if setting one of HEAD_ and TAIL_ to -1, no data of the file is available. It means when you load this file from PC, its size is 0 byte.

2. if setting HEAD_=1, TAIL_=1000, the current available data of the file will be at 1 ~ 1000 inside the volatile SRAM. It means when you load this file from PC, its size is 1000 bytes.

3. if setting HEAD_=10001, TAIL_=5000 : the current available data of the file will be at 10001 ~ 20000 and then continued with 1 ~ 5000 inside the volatile SRAM. It means when you load this file from PC, its size is 15000 bytes.

4. if setting HEAD_=1000, TAIL_=1000, no data of the file is available. It means when you load this file from PC, its size is 0 byte.

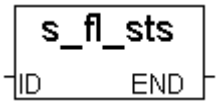
return:

Q_	**boolean**	TRUE: ok , FALSE: fail

Note: S_FL_INI should be called once before S_FL_AVL is called

Example: demo_40, demo_41, demo_42 & demo_44

S_FL_STS



Description:
Function Get file's Status, end byte No. that has been load by PC for the volatile SRAM

Arguments:

ID_ **Integer** File identifier No. (1 ~ 8)

return:

END_ **Integer** The end byte No. that has been load by PC

Not been load: -1
S256: 1 ~ 249,856
S512: 1 ~ 512,000
X607: 1 ~ 118,784
X608: 1 ~ 512,000

For ex.,

A file of ID_ = 1 is located at byte No. of 1 ~ 20000 , it can store max to 20000 bytes. And its current available data is setting at 1001 ~ 10000 inside the volatile SRAM.

1. If return END_ is -1, it means PC hasn't load it yet.

2. If return END_ is 10000 (Normally the value is equal to the current available ending byte No.), it means PC has load it from 1001 ~ 10000

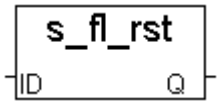
3. If return END_ is 8000, it means PC has load it from 1001 ~ 8000

Note:

1. S_FL_INI should be called first.
2. S_FL_RST can be called to reset the status to -1 (reset to "PC hasn't load it yet")

Example: demo_40, demo_41, demo_42 & demo_44

S_FL_RST



Description:
Function Reset file's Status to "Not been load by PC yet" for the volatile SRAM

Arguments:

ID_	**Integer**	File identifier No. (1 ~ 8)

return:

Q_	**Boolean**	TRUE: ok, FALSE: fail

Note:

1. S_FL_INI should be called first.
2. S_FL_STS can be called to get file's status

Example: demo_40, demo_41, demo_42 & demo_44

S_B_R

Description:
Function Read one boolean from the volatile SRAM

Arguments:

ADR_ Integer read which address, one Boolean occupy 1 byte.

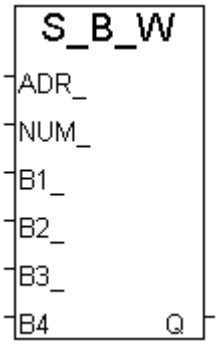
S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

return:

BOO_ Boolean The boolean value been read is 0=FALSE, not 0 = TRUE

Example: demo_40, demo_41, demo_42 & demo_44

S_B_W



Description:
Function Write max to 4 boolean to the volatile SRAM

Arguments:

ADR_ Integer start from which address, one Boolean occupy 1 byte.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

NUM_ Integer how many booleans to write, 0 ~ 4
B1_~B4_ Boolean the boolean value to write

return:

Q_ Boolean Ok: TRUE, Fail: FALSE

The boolean value will be stored is FALSE=0, TRUE=1

Example: demo_40, demo_41, demo_42 & demo_44

S_BY_R

Description:
Function Read one byte from the volatile SRAM

Arguments:
ADR_ Integer read which address, one Byte occupy 1 byte.

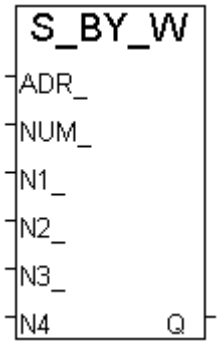
S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

return:

N_ Integer The byte value been read, 0 ~ 255

Example: demo_40, demo_41, demo_42 & demo_44

S_BY_W



Description:
Function Write max to 4 bytes to the volatile SRAM

Arguments:
ADR_ Integer start from which address, one Byte occupy 1 byte.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

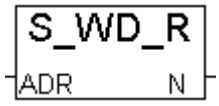
NUM_ Integer how many bytes to write, 0 ~ 4
N1_~N4_ Boolean the byte value (0-255) to write

return:

Q_ Boolean Ok: TRUE, Fail: FALSE

Example: demo_40, demo_41, demo_42 & demo_44

S_WD_R



Description:
Function Read one word from the volatile SRAM
Arguments:

ADR_ Integer read which address, one Word occupy 2 bytes.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

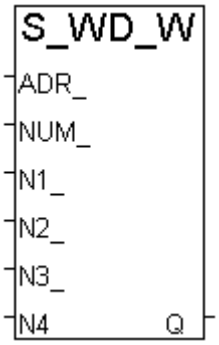
return:

N_ Integer The word value been read, -32768 ~ +32767

The word written in the SRAM is [Low byte] [High byte], for ex. a integer of 16#0102, it will be saved in the SRAM as [02] [01]

Example: demo_40, demo_41, demo_42 & demo_44

S_WD_W



Description:
Function Write max to 4 words to the volatile SRAM

Arguments:

ADR_ Integer start from which address, one Word occupy 2 bytes.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

NUM_ Integer how many words to write, 0 ~ 4
N1_~N4_ Integer the word value (-32768 ~ 32767) to write

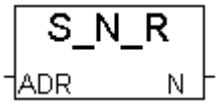
The word written in the SRAM is [Low byte] [High byte], for ex. a integer of 16#0102, it will be saved in the SRAM as [02] [01]

return:

Q_ Boolean Ok: TRUE, Fail: FALSE

Example: demo_40, demo_41, demo_42 & demo_44

S_N_R



Description:
Function Read one integer from the volatile SRAM

Arguments:
ADR_ Integer read which address, one Integer occupy 4 bytes.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

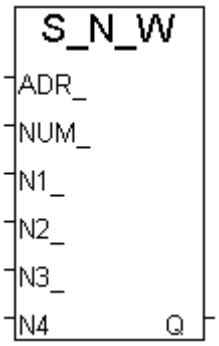
return:

N_ Integer The integer value been read, 32-bit, signed

The integer written in the SRAM is [Lowest byte] [2nd byte] [3rd byte] [High byte], for ex. a integer of 16#01020304, it will be saved in the SRAM as [04] [03] [02] [01]

Example: demo_40, demo_41, demo_42 & demo_44

S_N_W



Description:
Function Write max to 4 integers to the volatile SRAM

Arguments:
ADR_ Integer start from which address, one Integer occupy 4 byte.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

NUM_ Integer how many integers to write, 0 ~ 4
N1_~N4_ Integer the integer value (32-bit, signed) to write

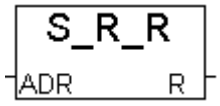
The integer written in the SRAM is [Lowest byte] [2nd byte] [3rd byte] [High byte], for ex. a integer of 16#01020304, it will be saved in the SRAM as [04] [03] [02] [01]

return:

Q_ Boolean Ok: TRUE, Fail: FALSE

Example: demo_40, demo_41, demo_42 & demo_44

S_R_R



Description:
Function Read one real value from the volatile SRAM

Arguments:
ADR_ Integer read which address, one Real value occupy 4 bytes.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

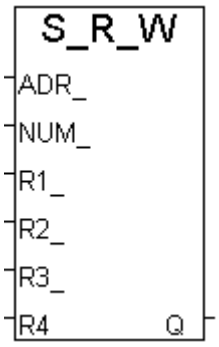
return:

R_ Real The real value been read, 32-bit float

The real value written in the SRAM is [Lowest byte] [2nd byte] [3rd byte] [High byte]. For ex. Real Value of 1.23 is consists of 4 bytes --> 16#A4 , 16#70 , 16#9D , 16#3F

Example: demo_40, demo_41, demo_42 & demo_44

S_R_W



Description:
Function Write max to 4 real values to the volatile SRAM

Arguments:
ADR_ Integer start from which address, one Real occupy 4 byte.

S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)

NUM_ Integer how many real values to write, 0 ~ 4
R1_~R4_ Real the real value (32-bit float) to write

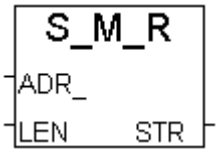
The real value written in the SRAM is [Lowest byte] [2nd byte] [3rd byte] [High byte]. For ex. Real Value of 1.23 is consists of 4 bytes --> 16#A4 , 16#70 , 16#9D , 16#3F

return:

Q_ Boolean Ok: TRUE, Fail: FALSE

Example: demo_40, demo_41, demo_42 & demo_44

S_M_R



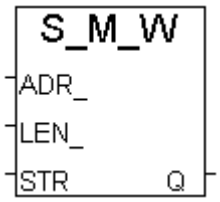
Description:
Function Read one string from the volatile SRAM
Arguments:
ADR_ Integer read which address.
S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)
LEN_ Integer Max length of the string to read, 0 ~ 255
return:
STR_ Message The string value been read

For ex., data in memory is 16#31, 16#32, 16#33, 16#34, 16#35, **0**, 16#37, 16#38, ...

LEN_=0 ----> STR_= '' (empty string)
LEN_=3 ----> STR_= '123'
LEN_=5 ----> STR_= '12345'
LEN_=6 ----> STR_= '12345'
LEN_=7 ----> STR_= '12345'
LEN_=100 ----> STR_= '12345'

Example: demo_40, demo_41, demo_42 & demo_44

S_M_W

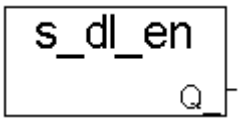


Description:
Function Write one string to the volatile SRAM

Arguments:
ADR_ Integer write to which address.
S256: 1 ~ 249,856 (1 ~ 16#3D000)
S512: 1 ~ 512,000 (1 ~ 16#7D000)
X607: 1 ~ 118,784 (1 ~ 16#1D000)
X608: 1 ~ 512,000 (1 ~ 16#7D000)
LEN_ Integer Max length of the string to write, 0 ~ 255.
STR_ IMessage the string value.

For ex.
LEN_=0 , STR='12345' ----> no data written
LEN_=1 , STR='12345' ----> 16#31 (1 byte written)
LEN_=3 , STR='12345' ----> 16#31, 16#32, 16#33 (3 bytes written)
LEN_=7 , STR='12345' ----> 16#31, 16#32, 16#33, 16#34, 16#35, 0, 0 (7 bytes written)
LEN_=100 , STR='12345' --> 16#31, 16#32, 16#33, 16#34, 16#35, 0, 0, 0, … (100 bytes written)
Return:
Q_ boolean Ok: TRUE, Fail: FALSE

S_DL_EN



Description:
Function Enable the download permission for PC to download data to the volatile SRAM

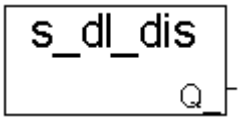
return:

Q_	**Boolean**	TRUE: ok, FALSE: fail

Note:
The default setting is "Disable". S_DL_EN sholud be called, then PC download data to the volatile SRAM is possible.

Example: demo_40, demo_41, demo_42 & demo_44

S_DL_DIS



Description:
Function Disable the download permission, so that PC can not download data to the SRAM

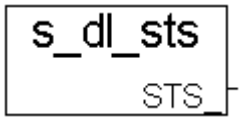
return:

Q_	**Boolean**	TRUE: ok, FALSE: fail

Note:
The default setting is "Disable". S_DL_EN sholud be called, then PC download data to the volatile SRAM is possible.

Example: demo_40, demo_41, demo_42 & demo_44

S_DL_STS



Description:
Function Get PC's Download Status for the volatile SRAM

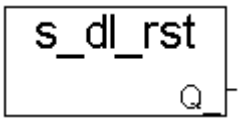
return:

STS_	**Integer**	-1:	No action,
		1:	PC is Downloading data to the SRAM now
		2:	download accomplishment

Note:
S_DL_RST can be called to reset the status to -1 (reset to "No action")

Example: demo_40, demo_41, demo_42 & demo_44

S_DL_RST



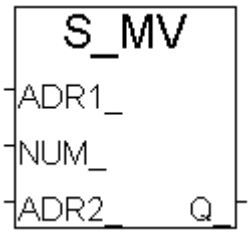
Description:
Function Reset the Download Status to "-1:No action" for the volatile SRAM

return:

Q_	**Boolean**	TRUE: ok, FALSE: fail

Example: demo_40, demo_41, demo_42 & demo_44

S_MV



Description:
Function **Move some bytes inside the volatile SRAM**
Arguments:

ADR1_ **Integer** destination start position

S256: 1 - 249856 (1 - 16#3D000)
S512: 1 - 512000 (1 - 16#7D000)
X607: 1 - 118784 (1 - 16#1D000)
X608: 1 - 512000 (1 - 16#7D000)

NUM_ **Integer** how many bytes to move, 0 - 512,000

ADR2_ **Integer** Move from which starting position

return:

Q_ **boolean** Ok: TRUE, Fail: FALSE

Example: demo_40, demo_41, demo_42 & demo_44