Retained Variables and Data Backup in I-8xx7 Modules

10.1: The Retained Variable

In some instances, data must be retained when the module's power is lost, and ideally, the data should be restored to its last value when the power powered back up. I-8xx7 & I-7188EG/XG controllers (**W-8xx7 doesn't support retained variable, however it supports file operation. Please refer to 10.5 – "Reading & Writting File"**) provide battery backup memories to fit such kind of applications. The battery used can provide the energy to keep the retained variables alive last for some years. It also can provide the energy for the Real-Time-Clock.

A maxinum of **six integers/reals** (signed 32-bit) and **sixteen Booleans** can be retained. If the amount of retained data is more, please refer to Section 10.3 – "Battery Backup SRAM" . If battery backup SRAM is found in the controller (8xx7: S256/S512, 7188EG/XG: X607/X608), the maxinum number of retained variables can be extend to 256 Boolean, 32 Timer and 256 Integer/Real.

To enable the retained function, click on "Retain" for each associated variable.

10.2: Data Backup To The EEPROM

Data can be stored into the EEPROM. The value will be always hold even the power is dead unless the value is updated. The EEPROM of I-8xx7, I-7188EG/XG & W-8xx7 controller can be read freely however can be written only about to 100,000 times.To read a value from the EEPROM, the following functions can be used.

To write a value to the EEPROM, should remove the protection of the EEPROM first and then write operation is possible. The following functions can be used.

The below two blocks can be used to Read/Save "real" value . To save a Real value to the EEPROM, use Real_Int to map the real value to an integer, and then use EEP_N_W to save this mapped integer. To read a Real value from EEPROM, use EEP_N_R to read it, and then use Int_Real to map this integer to an real value.

Bytes, words and integers will be stored to the same memory area in the EEPROM. Be careful to arrange their address before using the above write functions. There are total 1,512 bytes in the EEPROM memory area of the I-8xx7 & I-7188EG/XG, while much more in the W-8xx7.

For I-8xx7 & I-7188EG, the addressing No. of bytes is range from 1 to 1,512, while words is 1 to 756, and integers is 1 to 378. The following No. will use the same memory address in the EEPROM.

Byte 4n-3, 4n-2, 4n-1, 4n (* n = 1, 2, …378 *)
Word 2n-1, 2n
Integer n

Int_Real Map a long integer to a Real value.
Real_Int Map a Real value to a long integer.

EEP_EN Removes the protection of EEPROM
EEP_PR Set the protection of EEPROM
EEP_B_W Writes a boolean, up to 256 booleans can be stored.
EEP_BY_W Writes one byte, up to 1,512 bytes can be stored.
EEP_WD_W Writes one word (2 bytes, signed), up to 756 words can be stored.
EEP_N_W Writes one integer (4 bytes, signed), up to 378 integers can be stored.

EEP_B_R Reads one boolean
EEP_BY_R Reads one byte
EEP_WD_R Reads one word (2 bytes, signed)
EEP_N_R Reads one integer (4 bytes, signed)
For W-8xx7, the addressing No. of bytes is range from 1 to 14272, while words is 1 to 7136, and integers is 1 to 3568. The following No. will use the same memory address in the EEPROM.

When using the write functions, the EEPROM will be damaged if the write operation is more than 100,000 times. For example, the following program is dangerous since the EEPROM will be written once per cycle (normally, the cycle is about 2 to 60 ms depends on the application) .

However the following program is safe if Val is not changed frequently.

Each read / write operation in the EEPROM will consume a lot of CPU time of I-8xx7, I-7188EG/XG & W-8xx7 controller system. The following approximate time is for each function being called.

Recommend to read values from the EEPROM at one time when the I-8xx7, I-7188EG/XG & W-8xx7 is powered up, and then updated the associated address in the EEPROM when the value is changed. Please refer to a sample program in Chapter 11 – "**demo_17**". For those data which are frequently changed are not suitable to be stored in the EEPROM.

Byte 4n-3, 4n-2, 4n-1, 4n (* n = 1, 2, …3568 *)
Word 2n-1, 2n
Integer n

EEP_EN ~ 0.08 ms **EEP_PR** ~ 0.08 ms

EEP_B_R ~ 0.8 ms **EEP_B_W** ~ 6 ms
EEP_BY_R ~ 0.8 ms **EEP_BY_W** ~ 6 ms
EEP_WD_R ~ 1.5 ms **EEP_WD_W** ~ 12 ms
EEP_N_R ~ 2.9 ms **EEP_N_W** ~ 23 ms

```
(* ST program, Val, Old_Val declared as integers, TEMP declared as a boolean *)
IF  Val <> Old_Val  THEN
       TEMP := eep_n_w(1, Val);
      Old_Val := Val;
END_IF;
```

```
(* ST program, Val is declared as an integer, TEMP is declared as a boolean *)
TEMP := eep_n_w(1, Val);    (* dangerous *) .
```

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 1.18, and version 1.16 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 web site:

http://www.icpdas.com/products/8000/isagraf.htm

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 up to 8 files. Each file has a ID No. of 1 to 8 and a name of up 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.

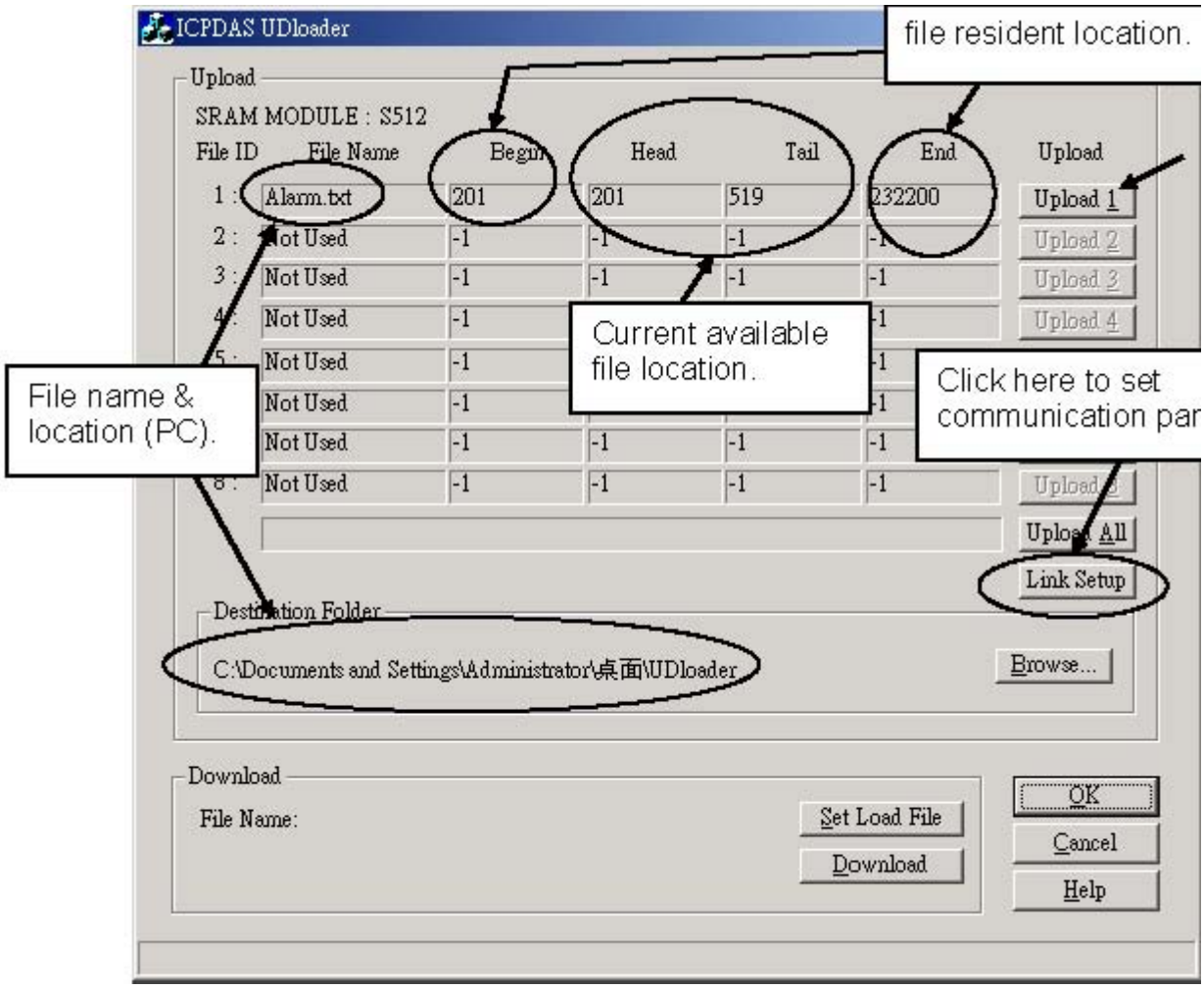
For ex, a file resides at (Begin, End) = (1, 20000)

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.



Click here to set communication parameters.

file resident location.

Current available file location.

File name & location (PC).

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_DIS, 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: up 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.

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

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 example:

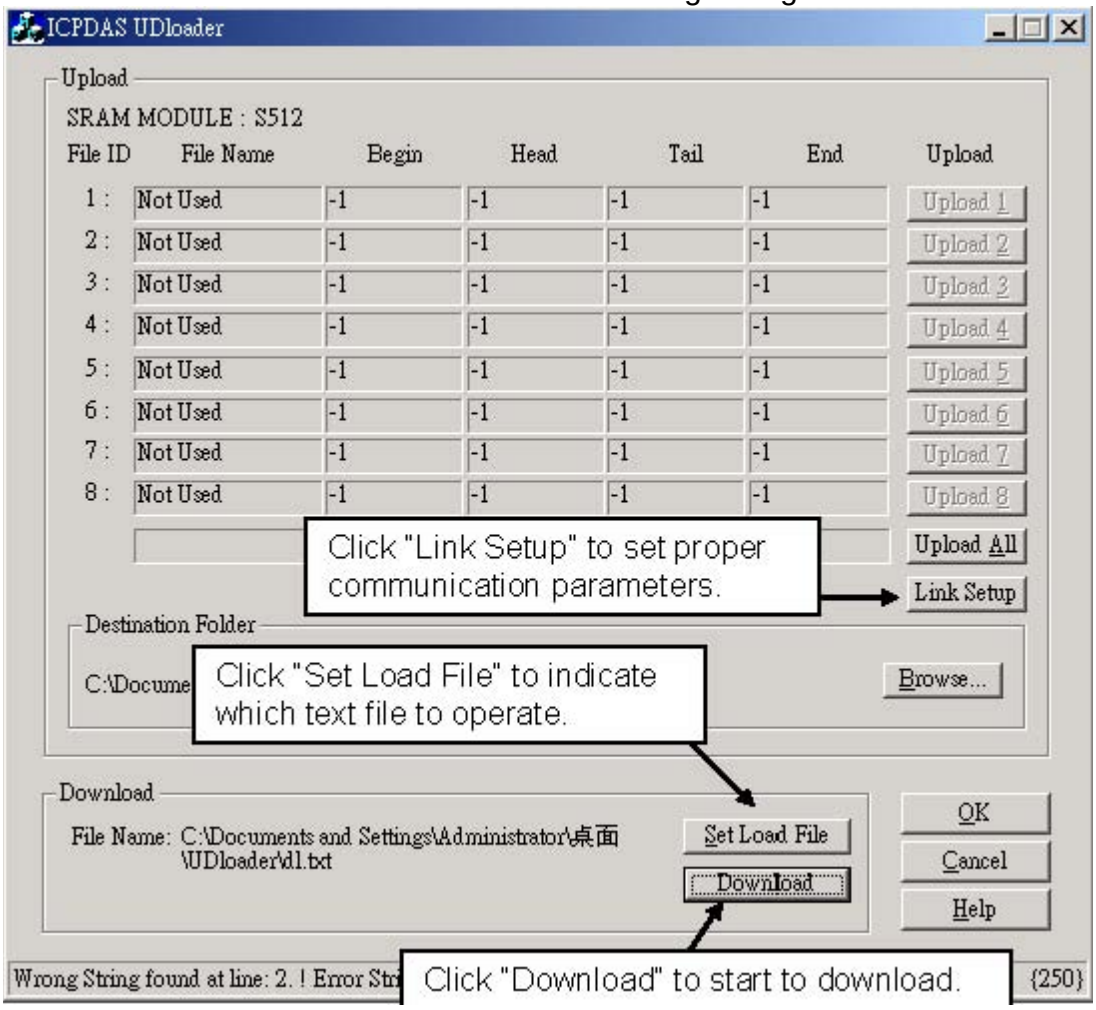
201 to download to the SRAM which staring from byte No. 201
‘Hello’ (10) (20) (30) (40) [-10000] {70000} ‘End’ data (total 18 bytes)

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

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.



Click “Download” to start to download.

Click “Set Load File” to indicate which text file to operate.

Click “Link Setup” to set proper communication parameters.

1
{1000} {250} {100} ‘sTART’

1 to download to the SRAM which staring from byte No. 1
(23) 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>

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

10.4: Using I-8073 - MultiMediaCard to store data

The I-8073 is not support by I-8xx7, I-7188EG/XG & W-8xx7.

10.5: Reading & Writing File

The W-8037/8337/8737 controller system support file operation however I-8xx7 & I-7188EG/XG doesn’t. W-8037/8337/8737 has a Compact Flah Disk with normal size of 128Mbytes (the size depends on the Compact Flash Disk been installed).

The following **ISaGRAF standard** functions are support by W-8xx7.

The following functions are support by W-8xx7.

The example programs for file operation reside at the Wincon CD-ROM:

- \napdos\isagraf\wincon\demo\ “wdemo_01.pia” & “wdemo_02.pia”
- ftp://ftp.icpdas.com./pub/cd/winconcd/napdos/isagraf/wincon/demo/

F_CREAT Creat an empty file for reading & writing .
F_SEEK Move file position to …
F_READ_B Read one byte (0 - 255) from a file .
F_WRIT_B Write one byte (0 - 255) to a file open with Write mode.
F_READ_W Read one Word (-32768 to +32767) from a file .
F_WRIT_W Write one byte (-32768 to +32767) to a file open with Write mode.
F_READ_F Read one float value(For ex. 123.45, -2.15E-03, …) from a file .
F_WRIT_F Write one float value to a file open with Write mode.

F_ROPEN Open an existing binary file in READ mode .
F_WOPEN Open an existing binary file in READ & WRITE mode .
F_CLOSE Close an open file
F_EOF Test if end-of-file has been reached
FA_READ Read one integer (4 bytes, signed) from a file.
FA_WRITE Write one integer (4 bytes, signed) to a file open with Write mode.
FM_READ Read one message (String) from a file.
FM_WRITE Write one message (String) to a file open with Write mode.