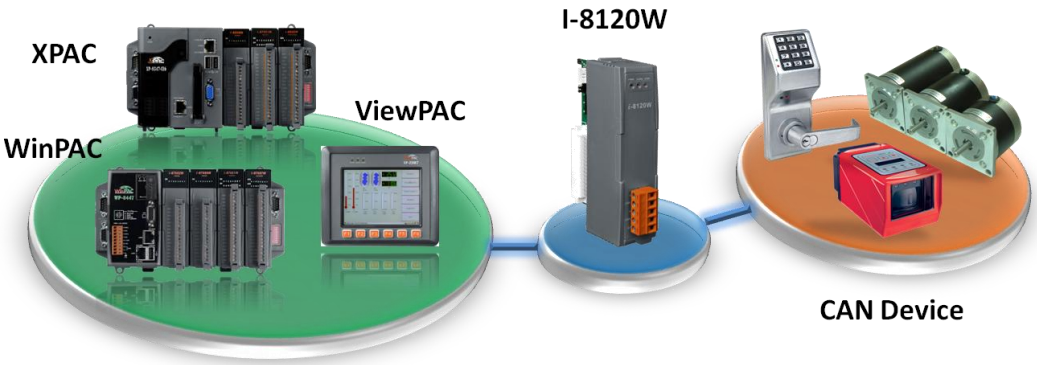
# How to use the ISaGRAF PAC and the I-8120W CAN module to read or control a CAN device?

ICP DAS ISaGRAF PAC - WP-8xx7, VP-25W7/23W7 and XP-8xx7-CE6 support the I-8120W intelligent CAN Bus module. The user can use the I-8120W to read/write data to the CAN device.




**I-8120W Features:**

- Follow ISO11898-2 specification and compatible with CAN specification 2.0 parts A and B
- NXP SJA1000T CAN controller with 16 MHz clock
- 2500 Vrms photo-isolation protection on CAN side
- Switch for 120 Ω terminator resistor of CAN bus
- Support 11-bit ID (CAN 2.0A) or 29-bit ID (CAN 2.0B) in the CAN frame
- Provide baud rate: 10 kbps, 20 kbps, 50 kbps, 125 kbps, 250 kbps, 500 kbps, 800 kbps and 1 Mbps
- 2048 records reception buffer and 256 records transmission buffer
- Timestamp of CAN message with ±1ms precision
- CPU: 80186, 80MHz
- Built-in RTC (Real Time Clock) and 8 KB DPRAM
- Support default firmware update and users to program user-defined firmware in I-8120W
- eVC++, VC6, VS 2005(VB.Net, C#.Net, VC++) demos and libraries are given
- C/C++ function libraries of firmware and demos are given

**Products Information:**

**ISaGRAF PAC:**
http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/isagraf/isagraf_tc.html

**I-8120W:**
http://www.icpdas.com/root/product/solutions/industrial_communication/fieldbus/can_bus/pac/i-8120w.html

## 1.1. Download/Update the ISaGRAF driver

The following ISaGRAF WinCE-based PAC support the I-8012W CAN module.

| ISaGRAF *WinCE-based* PAC | ISaGRAF Driver Version |
|---|---|
| XP-8xx7-CE6 | Ver. 1.58 or later |
| WP-8x47/8x37 | Ver. 1.79 or later |
| VP-25W7/23W7 | Ver. 1.70 or later |

Go to the website to download the latest driver: ISaGRAF Web Page > Download – Driver or http://www.icpdas.com/root/product/solutions/softplc_based_on_pac/isagraf/isagraf.html
(Unzip the file and follow instructions in the PDF to update the PAC driver.)

## 1.2. Download/Restore the ISaGRAF files

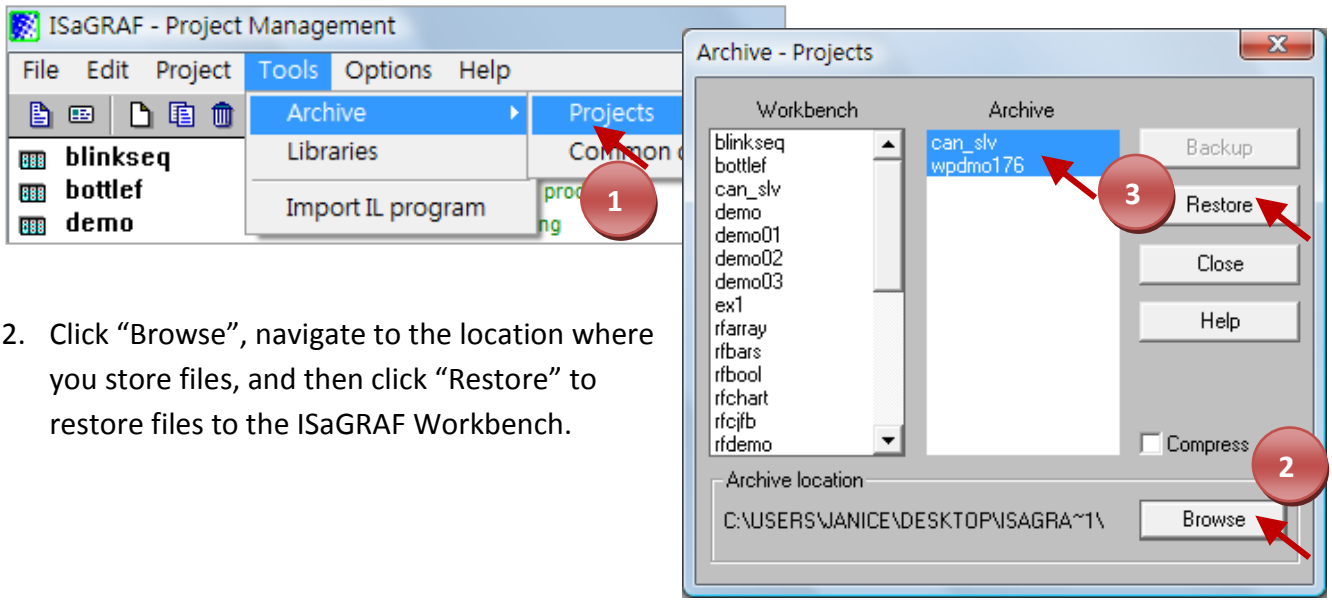
### 1.2.1. Download the ISaGRAF files

Go to ISaGRAF Web Page > Download – FAQ > FAQ-176 to download the "faq176_demo.zip" file. This file includes this document and the ISaGRAF files (Demo: wpdmo176.pia, can_slv.pia, IO boards: I_8120W.bia, C functions: can_by_w.uia, canstr_w.uia, C function blocks: can_r.fia ) 。

### 1.2.2. Restore the ISaGRAF files

**Restore the Demo Projects:**

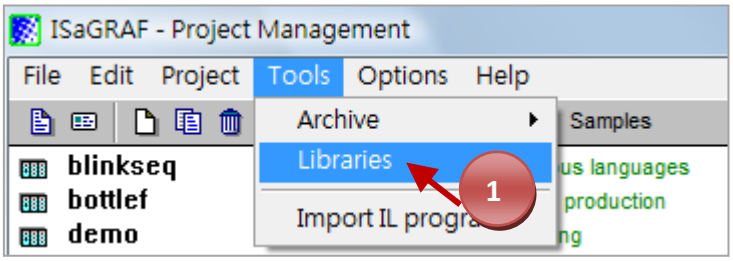
1. Click "Tools" > "Archive" > "Projects" to open the "Archive –Projects" window.



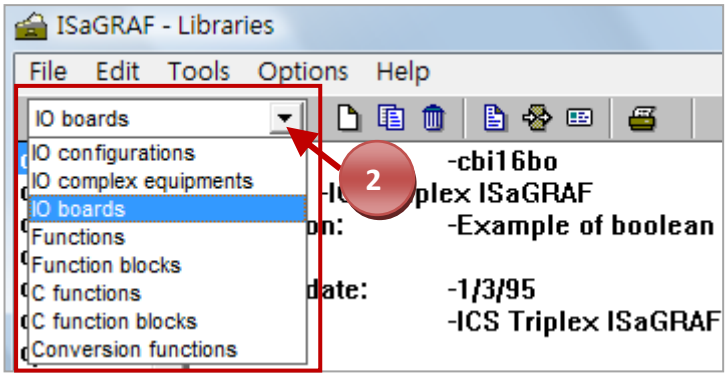

2. Click "Browse", navigate to the location where you store files, and then click "Restore" to restore files to the ISaGRAF Workbench.

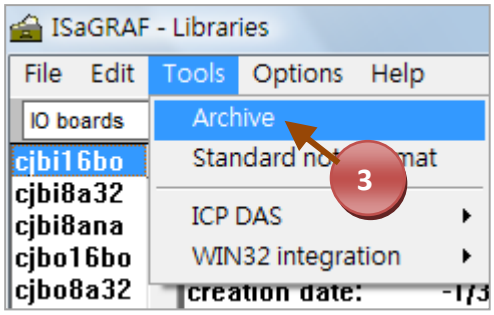
**Restore the Libraries:**

3. Click “Tools > Libraries” to open the “ISaGRAF – Libraries” window.

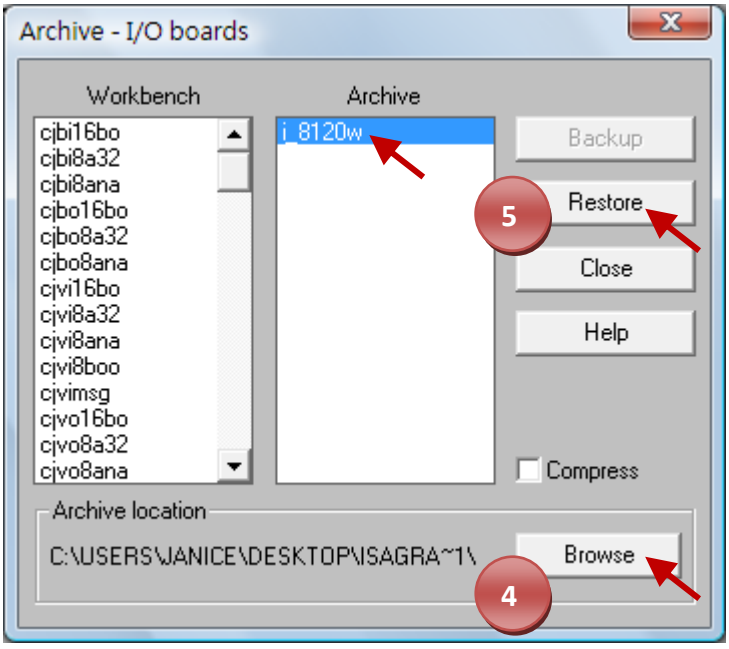



4. In the drop-down menu, choose the needed item (e.g., IO boards), and then click the menu bar “Tools” > “Archive” to open the “Archive” window.




5. Click “Browse”, navigate to the location where you store files, and then click “Restore” to restore file to the ISaGRAF Workbench.



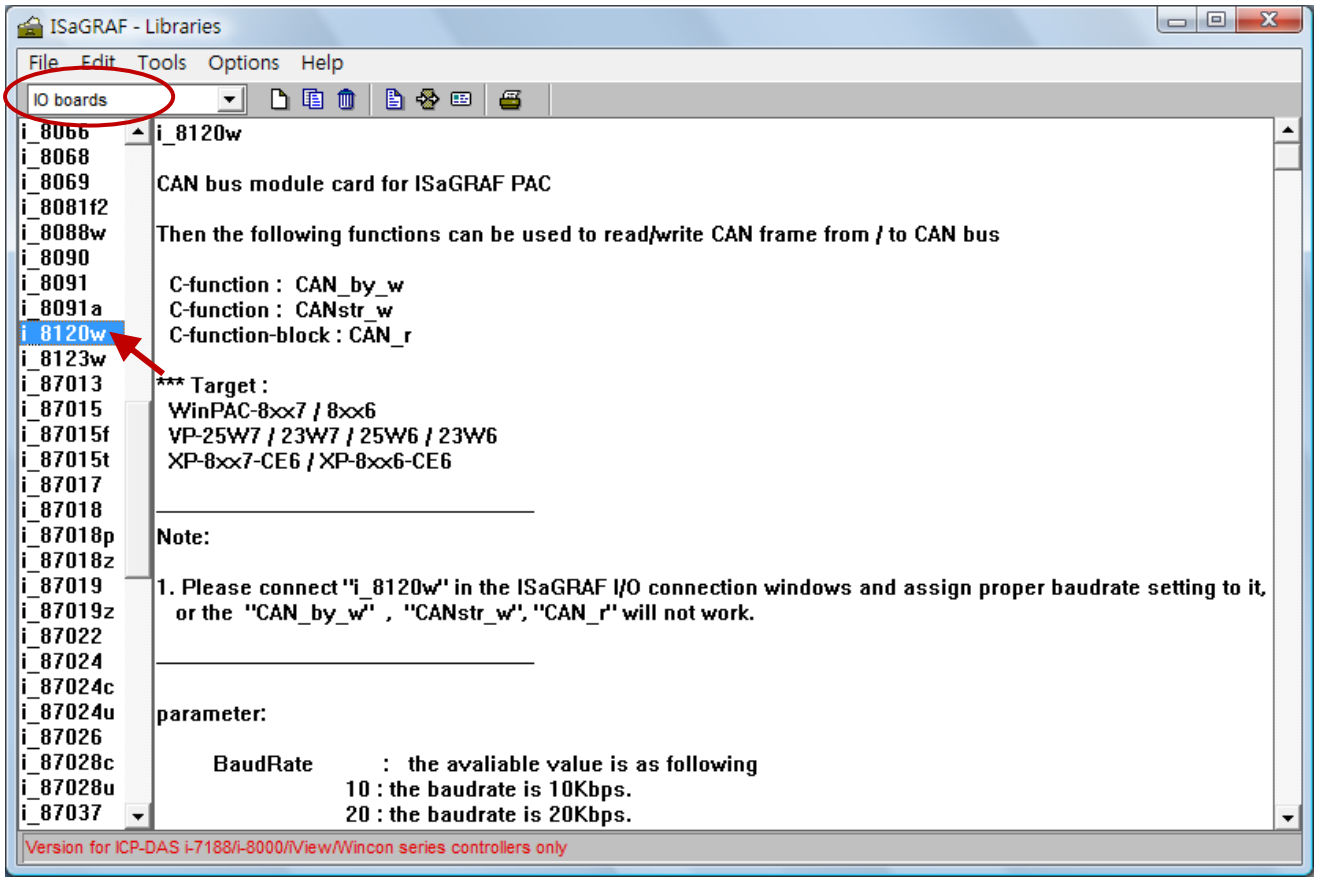

Using the same way to restore the others files.

C functions: can_by_w.uia, canstr_w.uia

C function blocks: can_r.fia

## 1.3. IO boards: I-8120W

After restoring the file, the user can click “i_8120w” in the “IO boards” option in the “ISaGRAF - Libraries” window to view detailed information.




### 1.3.1. Description of parameters

BaudRate: Valid values are as follows:

- 10 : the baud rate is 10 Kbps
- 20 : the baud rate is 20 Kbps
- 50 : the baud rate is 50 Kbps
- 125 : the baud rate is 125 Kbps
- 250 : the baud rate is 250 Kbps
- 500 : the baud rate is 500 Kbps
- 1000 : the baud rate is 1M bps

Acc_Code: 32-bit unsigned integer, acceptance code of CAN module.

Acc_Mask: 32-bit unsigned integer, acceptance mask of CAN module.

(See Section 1.3.2)

Analog Input channel:

Channel 1: Display the version of i8120.dll file.

Channel 2: Display the CAN Port number. If the I-8120W fails to initialize, this value will show 16#1FFFF.

Channel 3: Display the number of the received CAN frame.

Channel 4: Display the status of the CAN bus.

If this value is greater than "0"

bit 7 : Bus Status, 1 means bus-on, and 0 means bus-off.

bit 6 : Error Status, 1 means error, and 0 means OK.

bit 5 : Transmit Status, 1 means transmit, and 0 means idle.

bit 4 : Receive Status, 1 means receive, and 0 means idle.

bit 3 : Transmission Complete Status, 1 means complete, and 0 means incomplete.

bit 2 : Transmit Buffer Status, 1 means release, and 0 means locked.

bit 1 : Data Overrun Status, 1 means overrun, and 0 means absent.

bit 0 : Receive Buffer Status, 1 means full or not empty, and 0 means empty.

If this value is less than "0"

-23 : Cannot find the I-8120W module.

### 1.3.2. Basic concepts of the CAN bus frame

The I-8120W module support CAN specification 2.0 parts A (standard format) and B (extended format).

| ID | RTR | DLC | 8-byte Data |
|---|---|---|---|

The "**ID**" field is an identification number of the CAN frame.
If it is a CAN 2.0A frame, the ID field has 11 bits. So its value can be 0 ~ 7FF (Hex.).
While 29 bits for CAN 2.0B frame, so its value can be 0 to 1FFFFFFF (Hex.).

The "**RTR**" field has only 1 bit.
If its value is 1, it means the frame is using as "Remote-transmit requests", or called "Remote frame". It is for requesting the other CAN device to send proper data back. There is no Data field for "Remote frame".

If "**RTR**" is 0, then the frame is called "Standard frame". It is for sending data to other CAN devices. So "Standard frame" must have Data field.

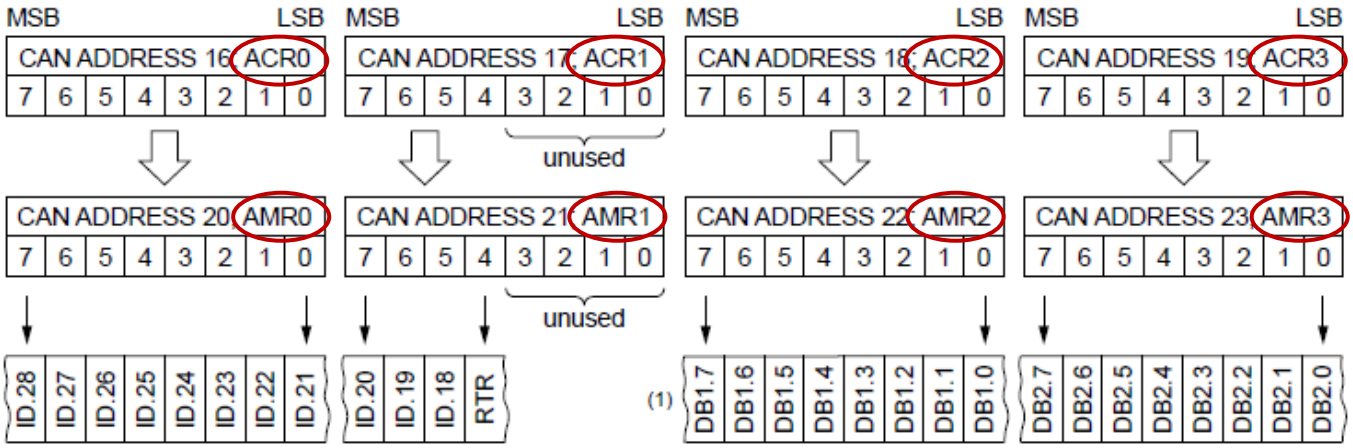
The "**DLC**" indicates the number of bytes in the Data field. Its value can be 0 to 8.

### 1.3.3. How does the I-8120W module determine whether to receive data?

The I-8120W module will compare all data bits of the "Acc_Code" and "Acc_Mask" parameters (Section 1.3.1) with the CAN frame (Section 1.3.2), and then decides whether to receive data. Therefore, we should understand the comparison way when using CAN specification 2.0A or 2.0B.

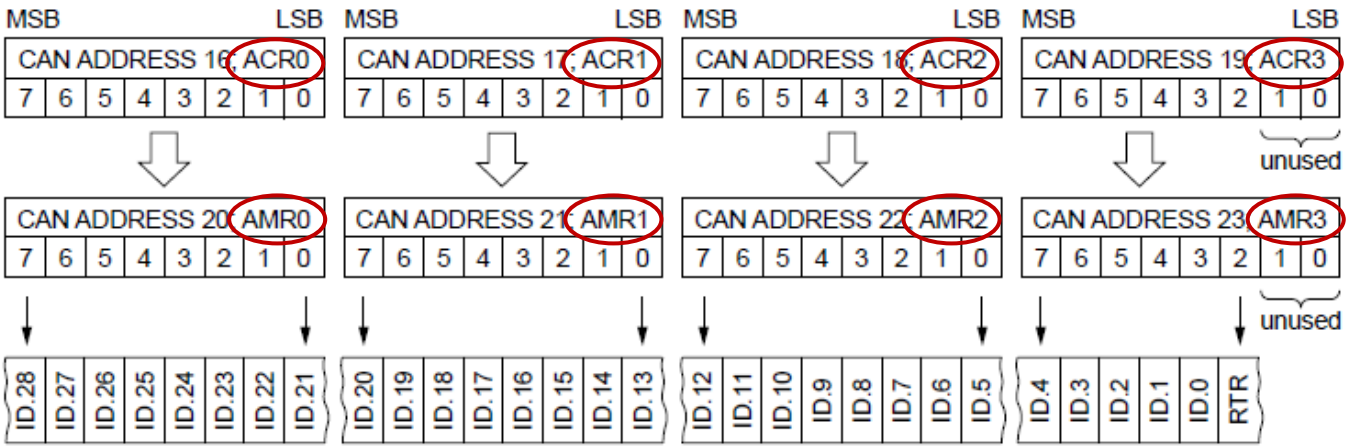
**CAN 2.0A (standard format):**

As the figure below, the "Acc_Code" (i.e., ACR0 ~ 3) and the "Acc_Mask" (i.e., AMR0 ~ 3) are 4 bytes data. The left-most two bytes, using their 11-bit to compare with "**ID**" (e.g., ID.18 ~ 28), 1-bit to compare with "**RTR**", and last 4-bit are unused. Moreover, it uses the rest two bytes to compare with the first two bytes of "**Data**".



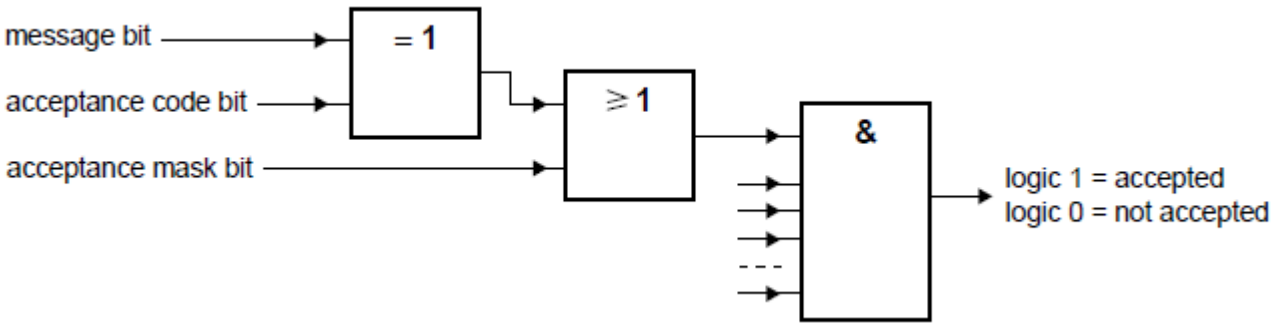

**CAN 2.0B (extended format)**

As the figure below, the "Acc_Code" (i.e., ACR0 ~ 3) and the "Acc_Mask" (i.e., AMR0 ~ 3) are 4 bytes data. It uses 29-bit to compare with "**ID**" (e.g., ID.0 ~ 28), 1-bit to compare with "**RTR**", and last 2-bit are unused.



To compare all data bits depends on the logic as the figure below.

1. If both values of the Message bit and “Acc_Code” bit are exactly equal, the result is “1”, else the result is “0”.
2. If the last result plus the “Acc_Mask” bit value is greater than or equal to “1”, the result is “1”.
3. When completing all data bits comparison, doing “AND” operations. If the result is “1”, the message will be accepted; else, it will not be accepted.




**Using CAN 2.0B (extended format) as an example (ID = 29 bit)**

When using CAN 2.0B, both the “Acc_Code” and “Acc_Mask” (32 bits) are mapping to the 29-bit “**ID**”, 1-bit “**RTR**”, and last 2-bit are unused. As the figure upon, we can know that only all “Acc_Mask” data bits are “1”, the message will be accepted no matter “Acc_Code” and “ID” values are the same or not. Therefore, the following ID Values from Array[0] to Array[2] are marked with “xxxx xxxx”, means no need to judge bit values.

| | Array[0] | Array[1] | Array[2] | Array[3] |
|---|---|---|---|---|
| Acc_Code | 00 (hex) | 00 (hex) | 00 (hex) | A0 (hex) |
| Acc_Mask | FF (hex) | FF (hex) | FF (hex) | 1F (hex) |
| ID bits | 28 ~ 21 | 20 ~ 13 | 12 ~ 5 | 4 ~ 0 |
| ID Values | xxxx xxxx | xxxx xxxx | xxxx xxxx | 101x x |

Seeing the first three bits in the Array[3], the “Acc_Mask” values are “0” that means only when “Acc_Code” and “ID” values are the same (i.e., 101), the message can be accepted. In addition, the rest two bits for “Acc_Mask” values are “1”, means no need to judge bit values.

| | Array[3] |
|---|---|
| Acc_Code | A0h = **101****0** **0**000 (2) |
| Acc_Mask | 1Fh = **000****1** **1**111 (2) |
| ID bits | 4 ~ 0 |
| ID Values | **101**x x |

## 1.4. Definitions of the CAN functions and function-block

The I-8120W supports the “CAN_R”, “CAN_BY_W”, and “CANSTR_W” functions to read/write the CAN frame on the CAN bus.

**CAN_R ( PORT_ ) : C-function block**

Test if any CAN frame coming.

**Note:** Remember to declare a FB instance when using ST program.

**Input parameters:**

| | | |
|---|---|---|
| **PORT_** | Integer | the value of channel 2 (i.e., CanPort) in the “i_8120w” I/O board. |

**Return values:**

| | | |
|---|---|---|
| **Q_** | Boolean | True: at least one CAN frame coming. False: no frame is coming. (Only when Q_ is “True”, the following return values are useful.) |
| **MODE_** | Integer | 0: using CAN 2.0A frame (ID has 11-bit).<br>1: using CAN 2.0B frame (ID has 29-bit). |
| **RTR_** | Integer | 0: using “Standard” frame (with 0 to 8 data bytes).<br>1: using “Remote” frame (without data byte). |
| **ID_** | Integer | ID in the current received frame. It can be 0 to 7FF (CAN2.0A frame) or 0 to 1FFFFFFF (CAN2.0B frame). |
| **DLC_** | Integer | the number of data bytes to receive (0 to 8). For example, if the “DLC_= 2”, it will get “BY1_” and “BY2_” data. |
| **BY1_ ~ BY8_** | Integer | 8 data byte, each one can be 0 to 255. (Only for “Standard” frame, not for “Remote” frame.) |
| **MSG_** | Message | the received String data. **Note:** if data contains byte value = 0, it will become the string end of this string. For ex, if the received frame has 8 bytes value in Hex format = 41, 42, 43, 4A, 0, 4B, 4C, 4D, then the MSG_ is 'ABCJ'. (MSG_ is only for standard frame, not for remote frame) |

**Note:** The user can click “can_r” in the “IO boards” option in the “ISaGRAF - Libraries” window to view detailed information. (See Section 1.3)

**CAN_BY_W ( PORT_ , MODE_ , RTR_ , ID_ , DLC_ , BY1_ , BY2_ , BY3_ , BY4_ , BY5_ , BY6_ , BY7_ , BY8 ) : 為 C-function**

Using CAN frame to send a max of 8 data bytes to the I-8120W CAN module (or CAN device).

**Note:** It needs to connect an "i_8120w" I/O board in the "ISaGRAF - I/O connection" window.

**Input parameters:**

| | | |
|---|---|---|
| **PORT_** | Integer | the value of channel 2 (i.e., CanPort) in the "i_8120w" I/O board. |
| **MODE_** | Integer | 0: using CAN 2.0A frame (ID has 11-bit).<br>1: using CAN 2.0B frame (ID has 29-bit). |
| **RTR_** | Integer | 0: using "Standard" frame (with 0 to 8 data bytes).<br>1: using "Remote" frame (without data byte). |
| **ID_** | Integer | the ID number for sending. It can be 0 to 7FF (CAN2.0A frame) or 0 to 1FFFFFFF (CAN2.0B frame). |
| **DLC_** | Integer | the number of data bytes to send (0 to 8). For example, if the "DLC_= 2" , it will get "BY1_" and "BY2_" data in a "Standard" frame. While using the "Remote" frame, it only send a request, the "DLC_= 0". |
| **BY1_ ~ BY8_** | Integer | 8 data byte, each one can be 0 to 255.<br>(Only for "Standard" frame, please set the unused byte as "0")。 |

**Return values:**

| | | |
|---|---|---|
| **Q_** | Boolean | True: send successfully. False: failed to send.<br>The failed reason can be:<br>(1) Input parameter is not correct, or<br>(2) The related CAN PORT_ is not open successfully, or<br>(3) Others. |

**Note:** The user can click "can_by_w" in the "IO boards" option in the "ISaGRAF - Libraries" window to view detailed information. (See Section 1.3)

**CANSTR_W ( PORT_ , MODE_ , ID_ , MSG_ ) : 為 C-function**

Using CAN frame to send a string to the I-8120W CAN module (or CAN device).

**Note:** If using the CANSTR_W( ) function, it will only send the “Standard” CAN frame. the user can use the CAN_BY_W( ) to send the “Remote” frame.

**Input parameters:**

| | | |
|---|---|---|
| **PORT_** | Integer | the value of channel 2 (i.e., CanPort) in the “i_8120w” I/O board. |
| **MODE_** | Integer | 0: using CAN 2.0A frame (ID has 11-bit).<br>1: using CAN 2.0B frame (ID has 29-bit). |
| **ID_** | Integer | the ID number for sending. It can be 0 to 7FF (CAN2.0A frame) or 0 to 1FFFFFFF (CAN2.0B frame). |
| **MSG_** | Message | the string for sending (max. 8 bytes). If the MSG_ is set to 'ClearCAN' (case-sensitive), all CAN frames sending from the specified “PORT_” will be cleared no matter what the "MODE_" and "ID_" setting are. |

**Note:** The “CANSTR_W” function can only send "standard" frame with a string (max. 8 characters). Each character cannot be “0”, because it indicates the end of the string. Using the “Com_BY_w” function if you want to send a "remote" frame or send data with value “0” (max. 8 bytes).

**Return values:**

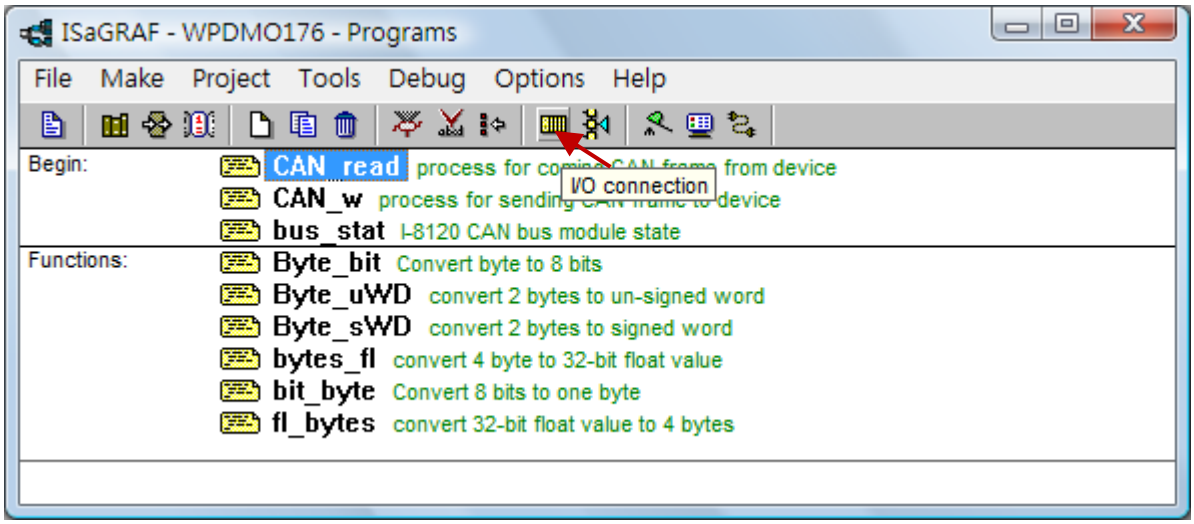
| | | |
|---|---|---|
| **Q_** | Boolean | True: send successfully. False: failed to send.<br>The failed reason can be:<br>(1) Input parameter is not correct, or<br>(2) The related CAN PORT_ is not open successfully, or<br>(3) Others. |

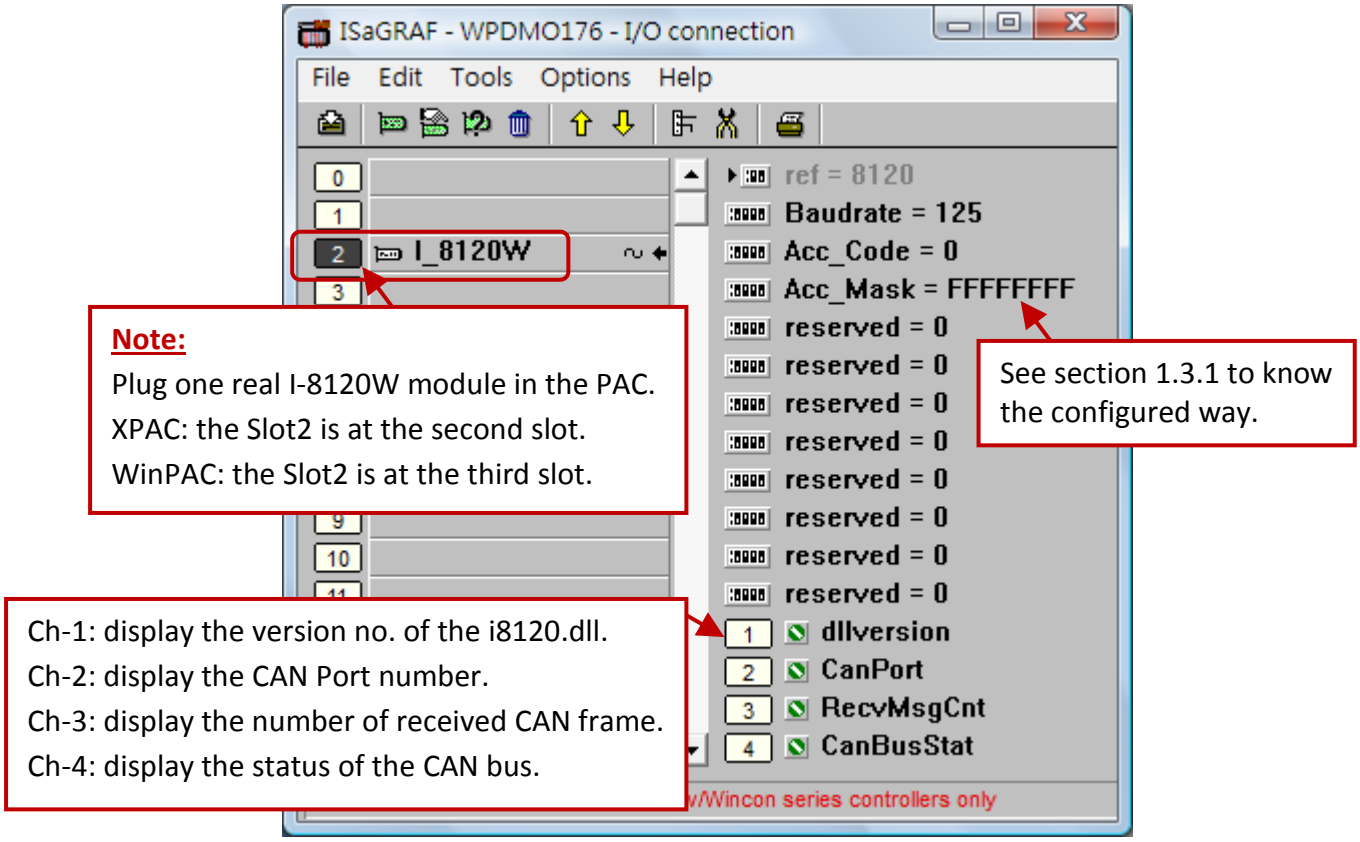
**Note:** The user can click “canstr_w” in the “IO boards” option in the “ISaGRAF - Libraries” window to view detailed information. (See Section 1.3) “

## 1.5. Introduction of the “WPDEMO176" demo

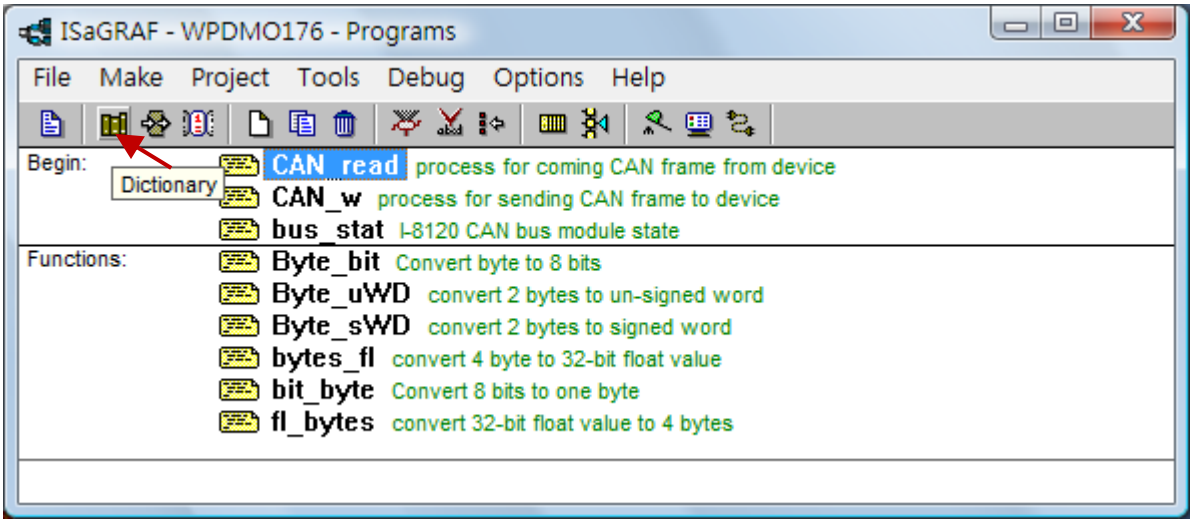
First, restore the “wpdemo176” demo file to the ISaGRAF in Section 1.2.2. This project includes several ST programs; you can double-click to open them and view the details. The following will describe “CAN_read” and “CAN_w” functions.

**Step 1:** Add the “I_8120W” in the Slot2 in the “I/O connection” window to establish a link between the software I/O board and the hardware I/O module.



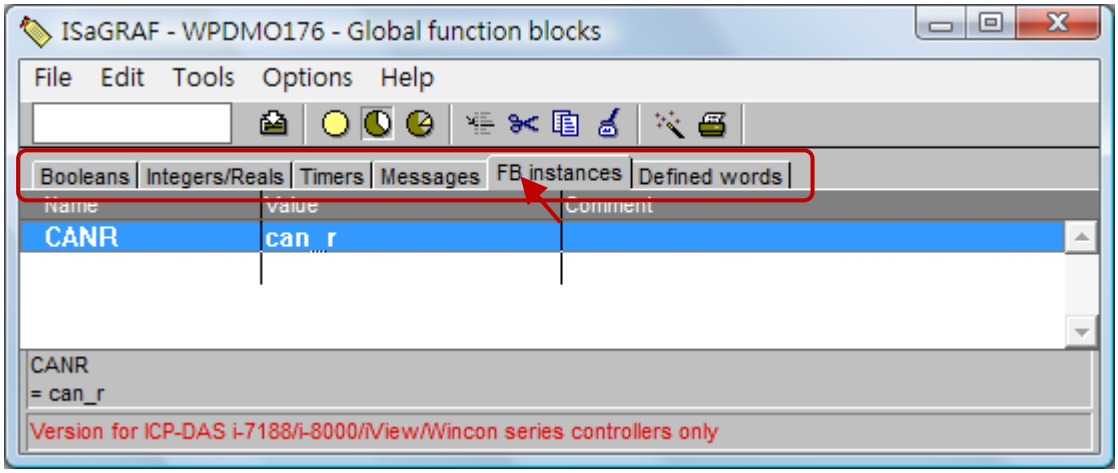



**Step 2:** the user can declare all needed variables in the "Dictionary" window.




You can click each tab to view all used variables in this project.

**Note:** When using the "CAN_R" function in the ST program, you must declare a FB instance.




**Step 3:** Writing a ST program - "CAN_read"

```
(* This demo project is to show how to use one I-8120W (CAN bus module) to
   communicate with the CAN device.

   Hardware Environment:

   1. Plug one i-8120W in the slot2 of the ISaGRAF PAC.
   2. Connect the CAN device to the I-8120W module.

   If you do not have CAN device yet, you could use I-7530 (RS-232 to CAN converter) to simulate
   the CAN device, as the follows.

   Wiring:          WP-8xx7 --- I-8120W ---- I-7530 --- PC
```

This demo uses one XP-8xx7-Atom-CE6 plus one I-8120W (baudrate: 125K, AccCode: 0, AccMask: 0xFFFFFFFF), and then connects to one I-7530 (115200, 8, N, 1, No Checksum, CAN 2.0A, 125K) that has connected to a PC for simulating.

PC can run i7530.exe (in the I-7530 CD-ROM) to send the following commands to simulate.

t18183EC5374841C83F22, CAN ID = 181
(DLC is 8 bytes: 3E C5 37 48 41 C8 3F 22 , Hex format)
(Two Real values: 0.385187 25.0308 )
(Then, you will see the ISaGRAF variable "float_val1" = 0.385187, "float_val2" = 25.0308 )

t1814312E3132452D3132, CAN ID = 181
(DLC is 4 bytes: 42 C9 8E D1 , Hex format)
(One Real value: 100.779 )
(Then, you will see the ISaGRAF variable "float_val1" = 100.779 )

t28163930312E3233, CAN ID = 281
(DLC is 6 bytes: 39 30 31 2E 32 33 , Hex format )
(String is 901.23: 9 0 1 . 2 3 )
(Then, you will see the ISaGRAF variable "Real_val" = 901.23)

t28123730, CAN ID = 281
(DLC is 2 bytes: 37 30 , Hex format )
(String is 70: 7 0 )
(Then, you will see the ISaGRAF variable "Real_val" = 70)

t381435363738, CAN ID = 381
(DLC is 4 bytes: 35 36 37 38 , Hex format )
(String is 5678: 5 6 7 8 )
(Then, you will see the ISaGRAF variable "int_val" = 5678)

t38152D35363738, CAN ID = 381
(DLC is 5 bytes: 2D 35 36 37 38, Hex format )
(String is -5678: - 5 6 7 8 )
(Then, you will see the ISaGRAF variable "int_val" = -5678)
*)

```
(*  Here we use the "RecvCnt_Max_one_time" to set the limited the number of CAN frames that
    can be handled within one PLC scan cycle *)

(*  The default value of "RecvCnt_Max_one_time" is 12 *)

if RecvMsgCnt > 0 then

        (* There is some CAN frame coming *)
        if RecvMsgCnt > RecvCnt_Max_one_time then
                Recv_Max := RecvCnt_Max_one_time;
        else
                Recv_Max := RecvMsgCnt;
        end_if;
else
        Recv_Max := 0;
end_if;

for i := 1 to Recv_Max do

        (*  Call C Function Block - "CAN_R", we have declared a FB instance of "CAN_R" ("CanR")
            in the ISaGRAF “Dictionary” window. *)
        CanR(CanPort);

        (* Get the result of calling "CanR( )" *)
        Can_Coming := CanR.Q_ ;         (* “True” means there is CAN frame coming *)

        if CanR.Q_ then
           Rx_count := Rx_count + 1;

           Can_Mode := CanR.MODE_ ;    (* The value of MODE_ is only returned as 0 or 1 *)
           Can_RTR    := CanR.RTR_ ;       (* The value of RTR_ is only returned as 0 or 1 *)
           Can_ID     := CanR.ID_ ;
           Can_DLC    := CanR.DLC_ ;       (* The value of DLC_ is only returned as 0 to 8 *)
           Can_By1    := CanR.BY1_ ;
           Can_By2    := CanR.BY2_ ;
           Can_By3    := CanR.BY3_ ;
           Can_By4    := CanR.BY4_ ;
           Can_By5    := CanR.BY5_ ;
           Can_By6    := CanR.BY6_ ;
           Can_By7    := CanR.BY7_ ;
           Can_By8    := CanR.BY8_ ;
           Can_str    := CanR.MSG_ ;
```

```
(* For CAN 2.0A frame, Can_Mode=0    *)
(* ---------------------------------- *)
if    Can_Mode = 0    then          (* The coming frame is CAN 2.0A frame *)
      if    Can_RTR = 1    then       (* Can_RTR = 1 : frame is "Remote" frame *)
      else
       (* Can_RTR = 0 : frame is "Remote" frame *)

       Case    Can_ID    Of
       16#181:  (* Real data in ID=16#181 has coming *)

         case Can_DLC of
          4:    (* Contain one real value *)
                float_val1 := bytes_fl(Can_By1, Can_By2, Can_By3, Can_By4);

          8:    (* Contain two real values *)
                float_val1 := bytes_fl(Can_By1, Can_By2, Can_By3, Can_By4);
                float_val2 := bytes_fl(Can_By5, Can_By6, Can_By7, Can_By8);
         end_case;

       16#281 :    (* String data with ID no. 16#281 is coming *)
         (* At least 1 byte data *)
         If    Can_DLC > 0    then
              (* Convert a String to one floating-point data *)
              tmp_float := str_real( Can_str ) ;
               if    tmp_float = 1.23E-20    then
                     (* If a format error occurs, str_real( ) will return 1.23E-20 *)
                    (* If a format error occurs, nothing will be handled *)
               else
                      Real_val := tmp_float ;      (* Store the result to "float_val *)
               end_if;
         End_if ;

        16#381 :    (* String data in ID=16#381 has coming *)
          (* At least 1 byte data *)
          If    Can_DLC > 0    then
          (* Convert a string to some integers and store them in the integer array No. 2*)
               tmp_val := ANA(Can_str) ;
          (* This demo assumes only 1 integer contained in this string, for example,
             '15003' or '-9123456'      , msg_n( ) returns 1: one integer converted
```

```
   (* If the received string is    '-A32' or '+-12'        , msg_n( ) returns -1: format error
      If the received string is '32,405' or '-12    83' , msg_n( ) returns 2: two integers converted *)
                        if    tmp_val <> 0    then
                      (*Retrieve this converted integer from address 1 of the integer array No. 2 *)
                            int_val := tmp_val ;
                        end_if;
                End_if ;

                (* If your CAN device has other frames coming, please put it here. *)
                (*
                    16#282 :
                                        .......
                *)

                  (* More ... *)
            End_case ;
        end_if; (* if    Can_RTR = 1    then *)
      else
      (* For CAN 2.0B frame , Can_Mode=1 *)
      (* ..................................................................... *)
      end_if; (* if    Can_Mode = 0    then *)
   end_if;
end_for;
```

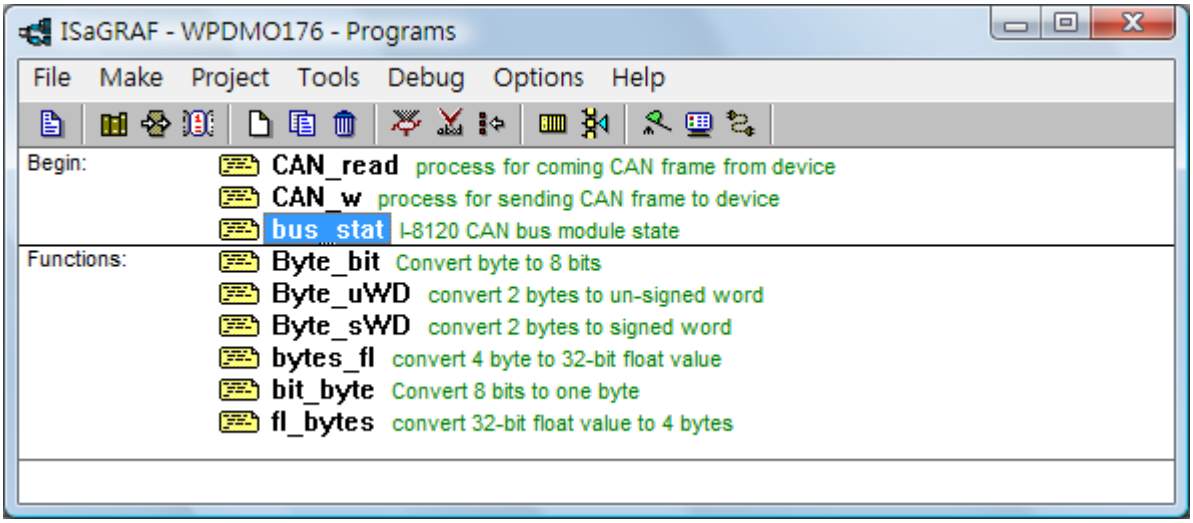
**Step 4:** Writing a ST program - “CAN_w”

```
(* Set "Send_en" as true to start to send CAN remote command *)
if send_en then
        send_en := false;
        (* Timer "TMR2" starts to count *)
        TStart(TMR2);
        (* Initialize the next tick count for sending CAN command *)
        TMR_next := TMR(Period2);
end_if;

(* Set "Send_disable" as “True” to stop sending CAN remote command *)
if send_disable then
        send_disable := false;
```

```
        (* Timer "TMR2" stops counting *)
        TStop(TMR2);
        (* Reset Timer "TMR2" as “0” *)
        TMR2 := T#0s;
end_if;

(* This demo uses "Period2" (default value is 20 ms) to send a CAN command.
   Users could set “Period2” as “0”, it means sending a CAN command every cycle. )

if Period2 <> 0 then
        if TMR2 >= TMR_next    then
                Send2 := True ;      (* Set it as “True” to trigger a sending action *)

                TMR_next := TMR_next + TMR(Period2);

                if    TMR2 > T#20h    then
                        TMR2 := T#0s;
                        TMR_next := TMR(Period2);
                end_if;
        end_if;
else
        Send2 := true; (* Set it as “True” to trigger a sending action *)
end_if;

If    Send2    then
        Send2 := False ;    (* Reset as “False” *)
        (*
          Step2 = 0 : Send CAN 2.0A, Remote command to ID = 16#181 to request data.
          Step2 = 1 : Send CAN 2.0A, Remote command to ID = 16#281 to request data.
          Step2 = 2 : Send CAN 2.0A, Remote command to ID = 16#381 to request data.
        *)
        Step2 := Step2 + 1;
        if Step2 = 3 then
           Step2 := 0;
        end_if;
       (* Operations in each step
           Boo_val := CAN_BY_W( Port_ , MODE_ , RTR_ , ID_ , DLC_ , BY1_ , ... , BY8_ )

           PORT_ :     Integer     The second channel value of "I_8120W" I/O board in the
                                   “IO Connection” window (i.e., CanPort).
```

```
        MODE_       Integer   0: Send it as CAN 2.0A frame (ID has 11 bits).
                              1: Send it as CAN 2.0B frame (ID has 29 bits).
        RTR_        Integer   0: Send it as standard frame (with BY1_ ~ BY8_).
                              1: Send it as remote frame (without BY1_ ~ BY8_).
        ID_         Integer   ID of the sending frame. CAN2.0A frame can be 0 ~ 7FF ;
                              CAN 2.0B frame can be 0 ~ 1FFFFFFF.
        DLC_        Integer   The number of data byte to send (0 ~ 8). For example,
                              if frame type is “Standard” frame and the DLC_ = 5, which
                              means to send BY1_ to BY5_. If frame type is “Remote” frame,
                              it will only send request frame without data byte (i.e., DLC_ = 0).
        BY1_ ~ BY8_ Integer   8 bytes data, each value can be 0 to 255.
                              (Only for standard frame, please set unused bytes as 0).
        return:     Boolean   True : Sending Ok ， False : Error.
    *)

    CASE  Step2  Of

     0 :
        TMP := CAN_BY_W( CanPort , 0 , 1 , 16#181 , 8 ,  0,0,0,0,  0,0,0,0 ) ;
       (* Send a CAN frame to the I-8120W module in the PAC’s Slot2 on the CAN bus to request
          data. 0: CAN 2.0A, 1: Remote frame, CAN ID = 16#381 (it depends on the CAN device),
          Data length: 8 bytes (it depends on the CAN device), and its data byte (BY1_ ~ BY8_)
          must set to “0”. Some CAN devices can automatically report data periodically, in this
          case, no need to send a "remote" frame.  *)
     1 :
       TMP := CAN_BY_W( CanPort , 0 , 1 , 16#281 , 8 ,  0,0,0,0,  0,0,0,0 ) ;

     2 :
       TMP := CAN_BY_W( CanPort , 0 , 1 , 16#381 , 8 ,  0,0,0,0,  0,0,0,0 ) ;

     (* If you have other "CAN" command to send, please put it here. *)
     (*
     3 :
       .....
     *)
    End_case ;

    if  TMP  then
        count_tx := count_tx + 1;
    end_if;
end_if;
```

**Step 5:** In the above steps, we have show you the “CAN_read” and the “CAN_w” programs. Then, you can double-click on other ST programs to view the details.



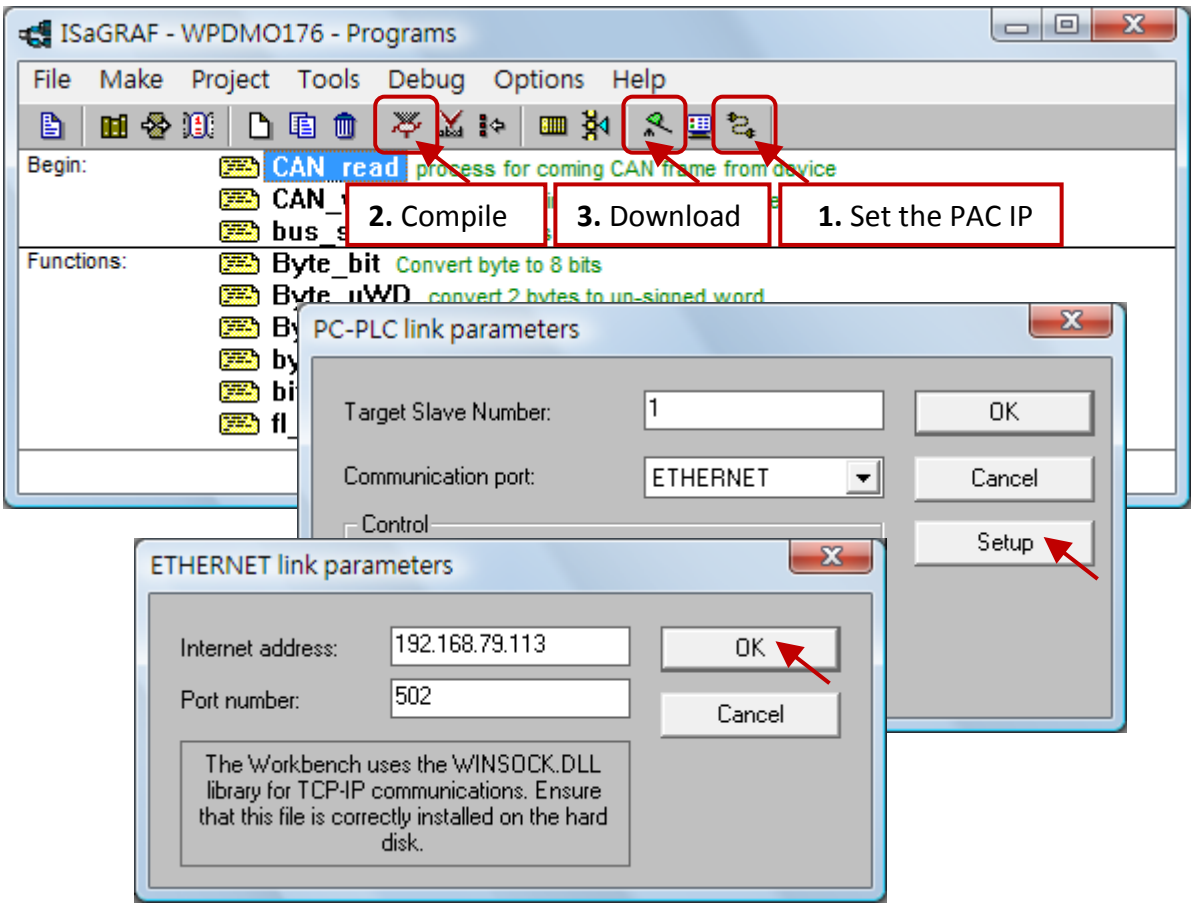

| Program Name | Purpose |
|---|---|
| CAN_read | Process for coming CAN frame device. |
| CAN_w | Process for sending CAN frame to device. |
| Bus_stat | Get the status of the I-8120W module. |
| Byte_bit | Convert 1 byte to 8 bit. |
| Byte_uWD | Convert 2 bytes to un-signed word. |
| Byte_sWD | Convert 2 bytes to signed word. |
| Bytes_fl | Convert 4 bytes to 32 bits float value. |
| bit_byte | Convert 8 bit to 1 byte. |
| fl_bytes | Convert 32 bits float value to 4 bytes. |

## 1.6. Testing

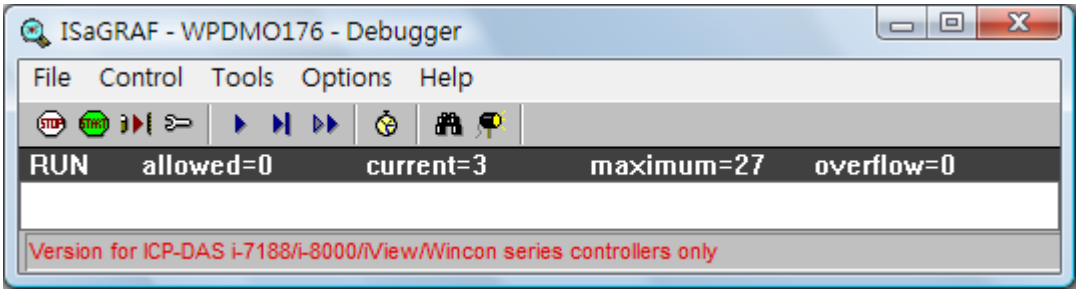
Before testing, you must download the ISaGRAF project (e.g., “wpdemo176”) to the ISaGRAF PAC.

### 1.6.1. Download the ISaGRAF program

1. Click the menu bar “Debug > Link setup” or  tool button to set the download IP (i.e., PAC IP).
2. Click the menu bar “Make > Make application” or  tool button to re-compile this project.




3. Click the menu bar “Debug > Debug” or  tool button to download the “wpdemo176”project. After downloading this project, the “Debugger” window will show as the figure below.

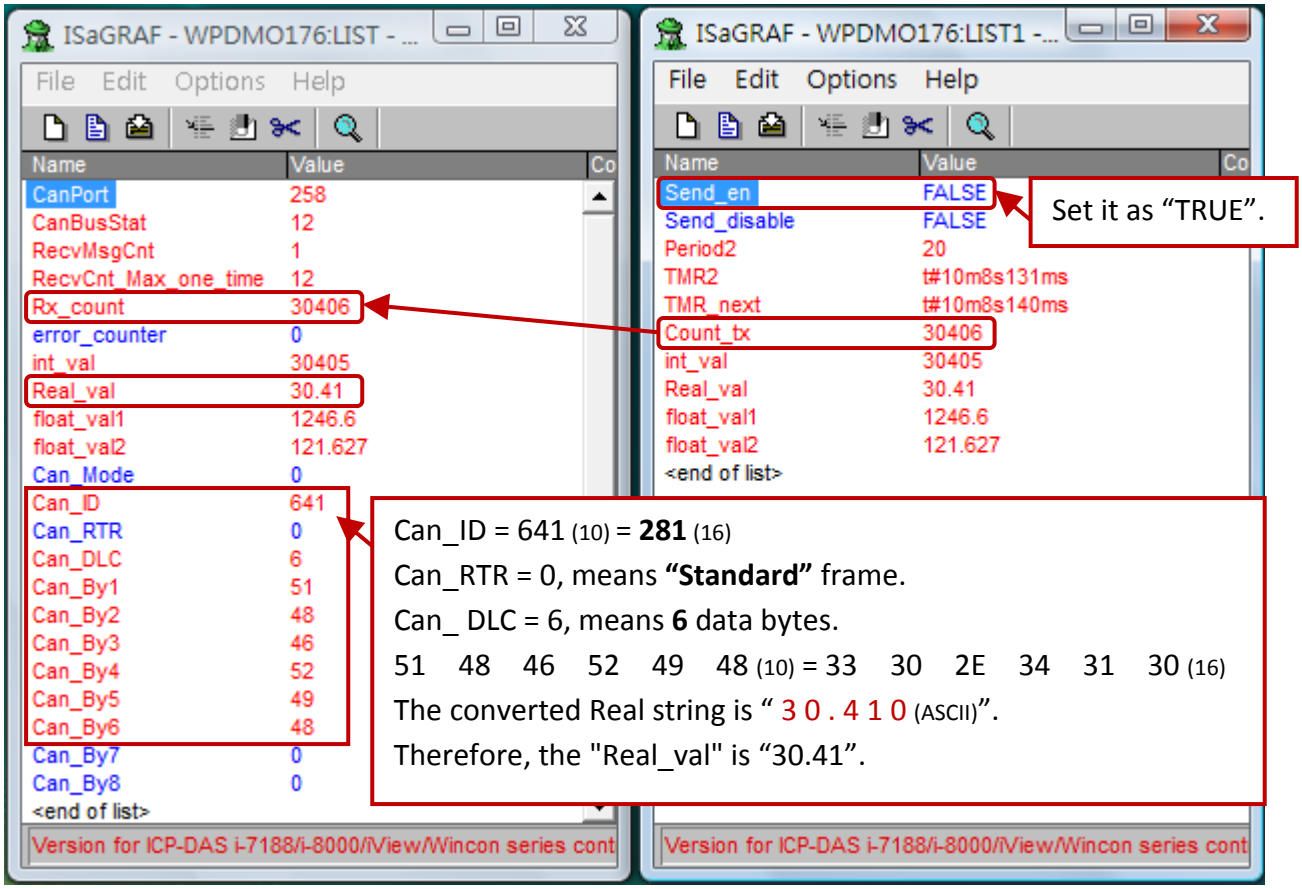


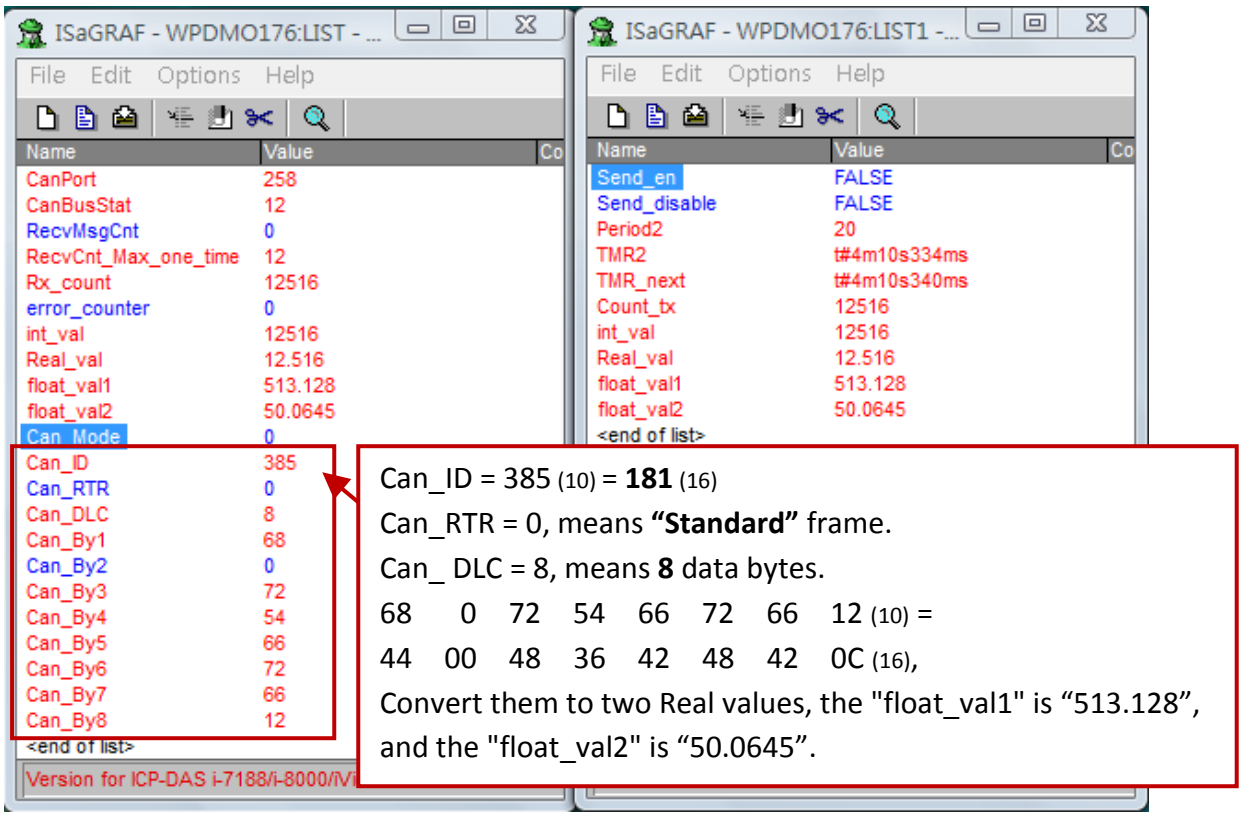
### 1.6.2. Test the ISaGRAF program

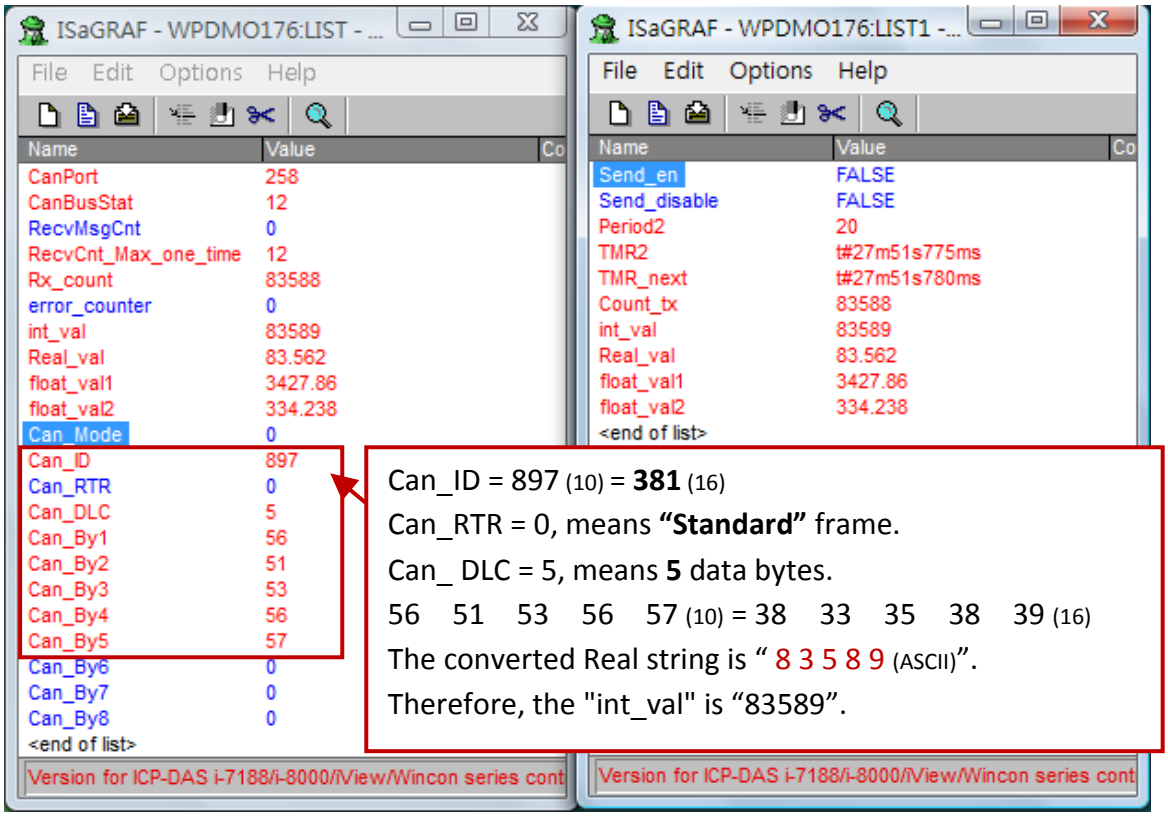
For convenient to test, we use two PACs and two I-8120W CAN modules which plugged in their slot2. In the table, the I-8120W module, which plugged in the WinPAC is used to simulate a CAN device.

| ISaGRAF PAC | Demo Program | I-8120W module | Wiring of modules |
|---|---|---|---|
| XPAC | wpdmo176.pia | Slot2 (the 2nd slot) | CAN_H to CAN_H<br>CAN_L to CAN_L |
| WinPAC | can_slv.pia | Slot2 (the 3rd slot) | |

After downloading the “wpdemo176” project to the ISaGRAF PAC, you can see two spy lists, mouse double-click on “Send_en” (Spy list1) to set it as “TRUE” (it will automatically set to “FALSE). Then, the program will starts the CAN bus communication. (Alternatively, you can refer Section 1.5 – Step 3 to conduct testing by using “i7530.exe”.)




ISaGRAF - WPDMO176:LIST - ...
File Edit Options Help
Name Value Co
CanPort 258
CanBusStat 12
RecvMsgCnt 0
RecvCnt_Max_one_time 12
Rx_count 12516
error_counter 0
int_val 12516
Real_val 12.516
float_val1 513.128
float_val2 50.0645
Can_Mode 0
Can_ID 385
Can_RTR 0
Can_DLC 8
Can_By1 68
Can_By2 0
Can_By3 72
Can_By4 54
Can_By5 66
Can_By6 72
Can_By7 66
Can_By8 12
<end of list>
Version for ICP-DAS i-7188/i-8000/iV
ISaGRAF - WPDMO176:LIST1 -...
File Edit Options Help
Name Value Co
Send_en FALSE
Send_disable FALSE
Period2 20
TMR2 t#4m10s334ms
TMR_next t#4m10s340ms
Count_tx 12516
int_val 12516
Real_val 12.516
float_val1 513.128
float_val2 50.0645
<end of list>
Can_ID = 385 (10) = 181 (16)
Can_RTR = 0, means "Standard" frame.
Can_ DLC = 8, means 8 data bytes.
68 0 72 54 66 72 66 12 (10) =
44 00 48 36 42 48 42 0C (16),
Convert them to two Real values, the "float_val1" is "513.128",
and the "float_val2" is "50.0645".


ISaGRAF - WPDMO176:LIST - ...
File Edit Options Help
Name Value Co
CanPort 258
CanBusStat 12
RecvMsgCnt 0
RecvCnt_Max_one_time 12
Rx_count 83588
error_counter 0
int_val 83589
Real_val 83.562
float_val1 3427.86
float_val2 334.238
Can_Mode 0
Can_ID 897
Can_RTR 0
Can_DLC 5
Can_By1 56
Can_By2 51
Can_By3 53
Can_By4 56
Can_By5 57
Can_By6 0
Can_By7 0
Can_By8 0
<end of list>
Version for ICP-DAS i-7188/i-8000/iView/Wincon series cont
ISaGRAF - WPDMO176:LIST1 -...
File Edit Options Help
Name Value Co
Send_en FALSE
Send_disable FALSE
Period2 20
TMR2 t#27m51s775ms
TMR_next t#27m51s780ms
Count_tx 83588
int_val 83589
Real_val 83.562
float_val1 3427.86
float_val2 334.238
<end of list>
Version for ICP-DAS i-7188/i-8000/iView/Wincon series cont
Can_ID = 897 (10) = 381 (16)
Can_RTR = 0, means "Standard" frame.
Can_ DLC = 5, means 5 data bytes.
56 51 53 56 57 (10) = 38 33 35 38 39 (16)
The converted Real string is " 8 3 5 8 9 (ASCII)".
Therefore, the "int_val" is "83589".