

Jun 12, 2000, Munich, Germany

by Robert Rostohar, Keil Elektronik GmbH    support.intl@keil.com    ++49 89 456040-0

The  $\mu$ Vision2 Debugger supports a simulator interface for implementing user defined peripherals. This interface is done via an **Advanced Generic Simulator Interface** called AGSI. The AGSI introduces a flexible and easy way for adding new user defined peripherals directly to  $\mu$ Vision2. It supports specific commands needed for simulation and maintenance of dialog pages associated with this peripherals.

To ease the development of a user defined peripheral, the AGSI and a configuration framework is provided in the SPeriDLL project. SPeriDLL, is a synonym for 'Sample Peripheral DLL'. It is a ready to run peripheral DLL with implemented 'A/D Converter from Analog Devices ADuC812' as a sample peripheral. It provides all the AGSI functions and also demonstrates the use of them by implementing the mentioned peripheral. The project consists of a MS Visual-C++ (6.0) project file and the following source files:

AGSI.h:	prototypes for the AGSI functions (do not modify !)
SPeriDLL.h:	main header file with various prototypes and definitions
SPeriDLL.cpp:	main file (created by AppWizard) provides peripheral setup code and simulation
PeriDialog.h:	header file (created by Class Wizard) for a modeless peripheral dialog
PeriDialog.cpp:	implementation file for a modeless peripheral dialog

Also a simple  $\mu$ Vision2 test project 'Single A/D conversion with ADuC812' is included in the file S812ADC.zip which shows how to include and test the implemented peripheral.

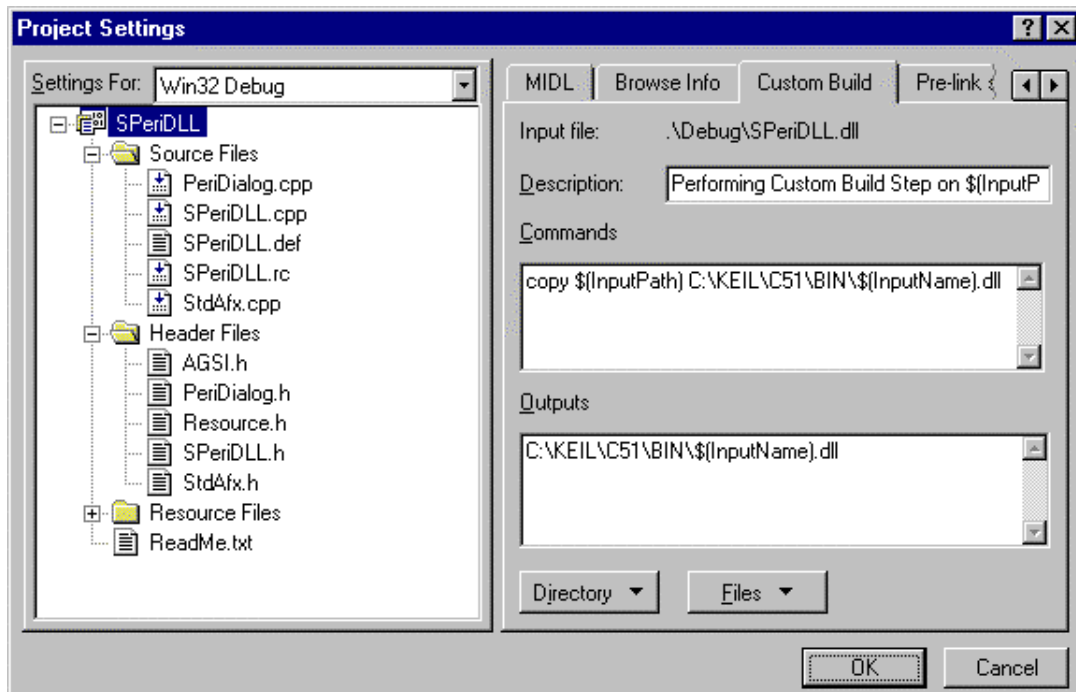
In order to develop a peripheral, knowledge about C/C++ programming and the MS Visual-C++ 6.00 Programming Environment is required.

## How to use the Sample Peripheral DLL

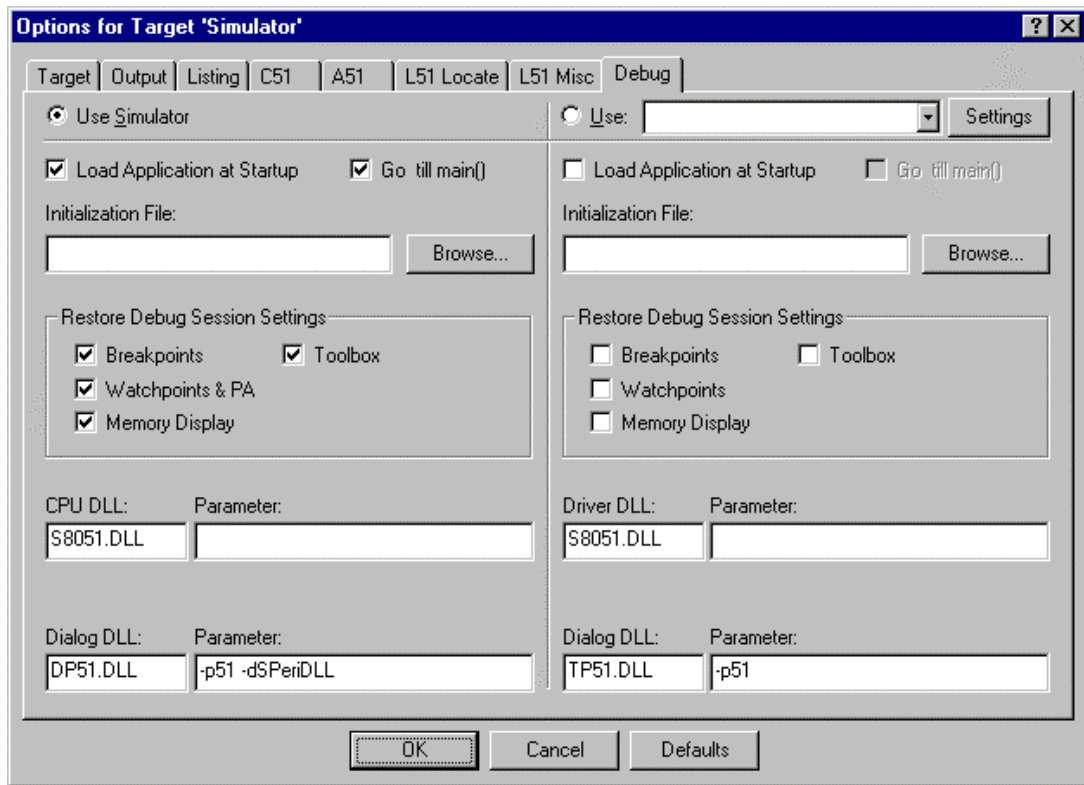
In order to use the Sample Peripheral DLL, you must perform the following steps:

- Install  $\mu$ Vision2 and the C51 Compiler on your machine.
- Create a folder such as D:\Src32\SPeriDLL\
- Unzip the file **SPeriDLL.zip** into the folder. Make sure that the 'use folder names' checkbox is checked since SPeriDLL uses some subfolders.
- Create a folder such as C:\Keil\C51\Examples\S812ADC\
- Unzip the file **S812ADC.zip** into the folder.

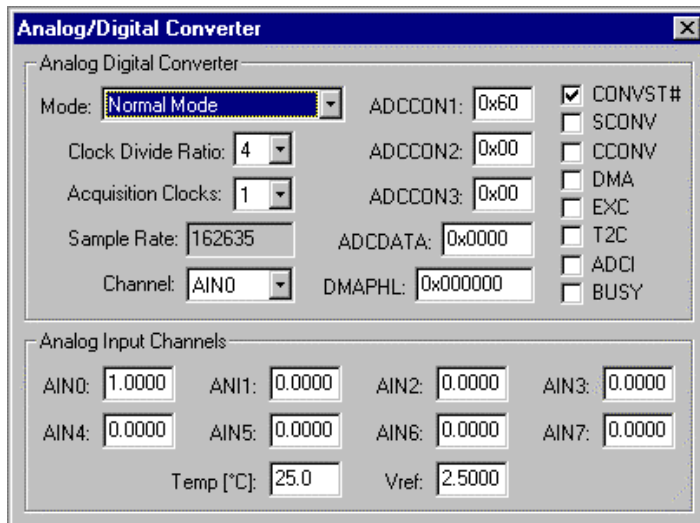
- Start Visual-C, select the 'SPeriDLL.dsw' project file.
- Select 'Project – Settings'. Click at the 'Debug' tab. Browse for the 'Executable for Debug session'. You need to select the file Uv2.Exe. It is normally in C:\Keil\Uv2 but this depends on where you have installed  $\mu$ Vision2.
- Then click at the 'Custom Build' tab and write in the 'Commands' window the command 'copy \$(InputPath) C:\KEIL\C51\BIN\\$(InputName).dll' and write in the 'Outputs' window the output file 'C:\KEIL\C51\BIN\\$(InputName).dll'. This step is required to automatically copy the created DLL after building it in the BIN subfolder of  $\mu$ Vision2 which is normally C:\KEIL\C51\BIN but depends on where you have installed  $\mu$ Vision2. If everything is right, then the dialog should look like this:



- After that, close the dialog.
- Select 'Build – Set active configuration', choose the SPeriDLL Win32 Debug configuration.
- Select 'Build – Rebuild All' to create the DLL.
- Run  $\mu$ Vision2 by pressing the F5 key. Select 'Project – Open Project', the Select Project dialog comes up. Select the 'S812ADC.uv2' project. It can be found in the folder that you have created and copied the project files into (normally C:\Keil\C51\Examples\S812ADC). Select 'Rebuild all target files' to build the project.
- Select 'Options for Target – Debug'. Enable loading of the SPeriDLL peripheral DLL by simply adding the parameter '-dSPeriDLL' to the parameter list of the 'Dialog DLL'. The parameter format for peripheral DLL's is '-dDLLName' (DLL name without extension). Make sure that the 'Use Simulator' radio button is checked. If everything is right, then the dialog should look like this:



- Close the dialog.
- Select 'Debug – Start/Stop Debug Session'. This will start the  $\mu$ Vision2 Debugger. It initializes and loads also our SPeriDLL.dll. In the 'Peripherals' menu a new item should be present with the label 'A/D Converter'. Click on this item to open the peripheral dialog which looks like this:



- Now you can single step through the code of the 'Single A/D conversion with ADuC812' sample and observe the behavior of the 'A/D Converter' peripheral and also other peripherals like 'Port 0', 'Port 2', 'Port 3' and 'Interrupt'.

Note that this sample program demonstrates functionality of only a small part of the A/D Converter.



---

## Implementing the peripheral DLL: Required Steps

In order to develop a peripheral DLL you should perform the following steps:

- Start MS Visual-C++ and create a new project 'MFC AppWizard (dll)'.
- Add prototypes and definitions for the AGSI and SFR's (Special Functions Registers) and VTR's (Virtual Registers) definitions to the main header file (\*.h).
- Write code for the peripheral initialization into the main file (\*.cpp). This includes AGSI setup (GetFunctionPointers), declaration of peripheral menu entries and associated dialogs (DefineAllMenuEntries), declaration of SFR's (DefineAllSFR), VTR's (DefineAllVTREG), Watches (DefineAllWatches) and Interrupts (DefineAllInterrupts). Write also code for peripheral reset - SFR's reset values (ResetPeripheral). All this functions are called from the function AgsiEntry() which must be exported by this peripheral DLL.
- Write functions for simulation of the peripheral into the main file (this functions are triggered by the defined watches). Include also prototypes of this functions.
- Create a peripheral dialog with the Resource Editor (if the dialog is required) and the associated header file (\*.h) and implementation file (\*.cpp) using the 'MFC ClassWizard'. Don't forget to set the 'Visible' property of the dialog and include the default buttons 'OK' and 'Cancel' and make them invisible (required for the behavior of the ESC and Enter keys). Change the default constructor for the dialog and add functions PeriDisp() – displays dialog, PeriUpdate() – updates display contents which calls function Update() and PeriKill() – closes the dialog. Add also a menu definition (AGSIMENU) and a dialog definition (AGSIDLGD).
- Write the code for updating the display contents into the Update() function in the dialog implementation file. This function is called automatically when an update is requested and is used to reflect the current state of the peripheral.
- Add functions for dialog control item's messages by using 'MFC ClassWizard'. Most frequently used messages are: ON\_BN\_CLICKED for Buttons, ON\_EN\_KILLFOCUS for Edit Boxes, ON\_CBN\_SELCHANGE for Combo Boxes ... Include also functions for the two invisible buttons 'OK' and 'Cancel'.
- Select 'Project – Settings'. Click at the 'Debug' tab. Browse for the 'Executable for Debug session' and select the file Uv2.Exe. It is normally in C:\Keil\Uv2 but this depends on where you have installed  $\mu$ Vision2.
- Rebuild your peripheral DLL. Then copy the DLL file to the BIN subfolder of  $\mu$ Vision2 which is normally C:\KEIL\C51\BIN but depends on where you have installed  $\mu$ Vision2 or use the 'Custom Build' within MS Visual-C++ and write the command that automatically copies the DLL after rebuild (see previous description in the 'Sample Peripheral DLL').
- Test your peripheral DLL by running  $\mu$ Vision2 (press the F5 key). Select a test project and enable loading of the implemented peripheral DLL by simply adding the parameter '-dDLLName' (DLL name without extension) to the parameter list of the peripheral DLL (see previous description in the 'Sample Peripheral DLL').
- If the implemented peripheral is running, switch into Release Mode and rebuild it. Then test the peripheral DLL again (don't forget to copy the 'Release DLL' file to the BIN subfolder of  $\mu$ Vision2).

---

## AGSI Functions

All type definitions and functions prototypes are included in header file **AGSI.h**.

```
DWORD AgsiEntry (DWORD nCode, void *vp);
```

$\mu$ Vision2 calls this function at the start of the debugging session first with `nCode=0` and the pointer `vp` being a pointer to the `CpuType` to match the CPU family of the current project with the CPU family of the peripheral DLL. Then this function is called again with `nCode=1` and the pointer `vp` being a pointer to `struct AGSICONFIG` to initialize the peripheral. The function is also called every time with `nCode=3` when a peripheral reset occurs (Reset CPU command) . It is also called with `nCode=2` at the end of the debugging session.

`AgsiEntry` gets the following `nCode` values:

```
nCode = AGSI_CHECK      = 0 // Check CPU Type
nCode = AGSI_INIT       = 1 // Initialize
nCode = AGSI_TERMINATE = 2 // Terminate
nCode = AGSI_RESET      = 3 // Reset
nCode = AGSI_PREPLL     = 4 // Before PLL change
nCode = AGSI_POSTPLL    = 5 // After PLL change
```

The function should return `TRUE(1)` if completed successfully or `FALSE(0)` if an error occurred.

Note: This function must be exported by the implemented DLL.

```
BOOL AgsiDefineSFR(const char* pszSfrName, AGSIADDR dwAddress,
                  AGSITYPE eType, BYTE bBitPos);
```

This function is used to declare a SFR (Special Function Register) or a SFR bit (bit addressable).

Input parameters:

<code>pszSfrName</code>	Name of the SFR
<code>dwAddress</code>	Address of the SFR
<code>eType</code>	Type of the SFR (byte, word or bit)
<code>bBitPos</code>	Bit position within SFR (for SFR bit)

Return value: `TRUE` if successful otherwise `FALSE`

Note: This function is allowed to call only during initialization process.

Examples:

```
AgsiDefineSFR("IE", 0xA8, AGSIBYTE, 0); // IE
AgsiDefineSFR("EA", 0xA8, AGSIBIT, 7); // EA bit in IE
```

```
AGSIVTR AgsiDefineVTR(const char* pszVtrName, AGSITYPE eType,
                    DWORD dwValue);
```

This function is used to declare a VTR (Virtual Register).

Input parameters:

<code>pszVtrName</code>	Name of the VTR
<code>eType</code>	Type of the VTR (char, word, long or float)
<code>dwValue</code>	Initial Value of the VTR

Return value: VTR handle if successful otherwise `NULL`

---

Note: This function is allowed to call only during initialization process.

Examples:

```
hXTAL = AgsiDefineVTR("XTAL", AGSIVTRLONG, 0x00B71B00); // 12MHz
hVREF = AgsiDefineVTR("VREF", AGSIVTRFLOAT, 0x40200000); // 2.5V
```

**BOOL AgsiDeclareInterrupt(AGSIINTERRUPT \*pInterrupt);**

This function is used to declare an interrupt as described in the following AGSIINTERRUPT structure:

```
typedef struct {
    AGSIADDR    vec;        // interrupt vector address
    char        *mess;     // interrupt name
    AGSIADDR    msfr;      // interrupt mode sfr
    WORD        mmask;     // interrupt mode bit mask
    const char  *mname;    // name of interrupt mode bit
    AGSIADDR    rsfr;      // interrupt request sfr
    WORD        rmask;     // interrupt request bit mask
    const char  *rname;    // name of interrupt request bit
    AGSIADDR    esfr;      // interrupt enable sfr
    WORD        emask;     // interrupt enable bit mask
    const char  *ename;    // name of interrupt enable bit
    AGSIADDR    p0sfr;     // interrupt priority 0 sfr
    WORD        p0mask;    // interrupt priority 0 bit mask
    const char  *pname;    // name of interrupt priority bit
    AGSIADDR    p1sfr;     // interrupt priority 1 sfr
    WORD        p1mask;    // interrupt priority 1 bit mask
    WORD        pwl;       // priority within level (1 - lowest priority)
    WORD        auto_reset; // reset interrupt request flag on interrupt entry
} AGSIINTERRUPT;
```

Input parameter: pInterrupt Pointer to the AGSIINTERRUPT

Return value: TRUE if successful otherwise FALSE

Note: This function is allowed to call only during initialization process.

Example:

```
#define TCON 0x88
#define IE 0xA8
#define IP 0xB8
#define IPH 0xB7

AGSIINTERRUPT ExtInt0 = { // External Interrupt 0
    0x0003, "P3.2/Int0", TCON, 0x01, "IT0", TCON, 0x02, "IE0", IE, 0x01, "EX0", IP, 0x01, "Pri",
    IPH, 0x01, 8, 1
};

AGSIINTERRUPT Timer0Int = { // Timer 0 Interrupt
    0x000B, "Timer 0", 0, 0, "", TCON, 0x20, "TF0", IE, 0x02, "ET0", IP, 0x02, "Pri",
    IPH, 0x02, 6, 1
};

Agsi.DeclareInterrupt(&Timer0Int);
Agsi.DeclareInterrupt(&ExtInt0);
```

**BOOL AgsiSetWatchOnSFR(AGSIADDR SFRAddress, AGSICALLBACK  
pfnReadWrite,AGSIACCESS eAccess);**

This function is used to set a watch on SFR access.

Input parameters:

SFRAddress	Address of the SFR
pfnReadWrite	Pointer to a function that is called on SFR access
eAccess	Access type (Read, Write or ReadWrite)

---

Return value: TRUE if successful otherwise FALSE

Note: This function is allowed to call only during initialization process.

Example:

```
#define TCON 0x88
#define TL1 0x8B
#define TH1 0x8D
#define TCON 0x88

static void timer1(void) {
// watch function implementation
}

AgsiSetWatchOnSFR(TH1, timer1, AGSIREADWRITE);
AgsiSetWatchOnSFR(TL1, timer1, AGSIREADWRITE);
AgsiSetWatchOnSFR(TCON, timer1, AGSIWRITE);
```

**BOOL AgsiSetWatchOnVTR(AGSIVTR hVTR, AGSICALLBACK pfnReadWrite, AGSIACCESS eAccess);**

This function is used to set a watch on VTR access.

Input parameters:

hVTR	Handle of previously defined VTR
pfnReadWrite	Pointer to a function that is called on VTR access
eAccess	Access type (Read, Write or ReadWrite)

Return value: TRUE if successful otherwise FALSE

Note: This function is allowed to call only during initialization process.

Example:

```
static void timer1(void) {
// watch function implementation
}

hPORT3 = AgsiDefineVTR("PORT3", AGSIVTRCHAR, 0xFF); // Port 3 pins

AgsiSetWatchOnVTR(hPORT3, timer1, AGSIWRITE);
```

**BOOL AgsiSetWatchOnMemory(AGSIADDR StartAddress, AGSIADDR EndAddress, AGSICALLBACK pfnReadWrite, AGSIACCESS eAccess);**

This function is used to set a watch on memory range access.

Input parameters:

StartAddress	Start Address of Memory range
EndAddress	End Address of Memory range
pfnReadWrite	Pointer to a function that is called on Memory range access
eAccess	Access type (Read, Write or ReadWrite)

Return value: TRUE if successful otherwise FALSE

Note: This function is allowed to call only during initialization process.

Example:

```
AgsiSetWatchOnMemory(0x0200, 0x02FF, AGSIWRITE); // Watch on Write to Memory range
```

---

```
AGSITIMER AgsiCreateTimer(AGSICALLBACK pfnTimer);
```

This function is used to create a timer watch.

Input parameters:

`pfnTimer`                      Pointer to a function that is called when timer expires

Return value: Timer handle if successful otherwise NULL

Note: This function is allowed to call only during initialization process.

Example:

```
static void AdcCompleted(void) {  
// Timer function implementation  
}  
  
AGSI Timer;  
Timer = Agsi.CreateTimer(AdcCompleted);
```

```
BOOL AgsiDefineMenuItem(AGSIMENU *pDym);
```

This function is used to define a new menu item under the 'Peripherals' menu in  $\mu$ Vision2 and the associated dialog. The menu item is described in the following AGSIMENU structure:

```
#define AGSIMENU struct AgsiDynaM  
struct AgsiDynaM {                      // Menu item data structure  
  int                    nDelim;            // Menu template delimiter  
  char                  *szText;         // Menu item text  
  void    (*fp) (AGSIMENU *pM);         // create/bring DlgtoTop function  
  DWORD                nID;             // uv2 assigned ID_xxxx  
  DWORD                nDlgId;         // Dialog ID  
  AGSIDLGD             *pDlg;         // link to dialog attributes  
};  
  
// nDelim:  1 := normal Menu entry  
//           2 := Popup-Entry (nested submenu)  
//          -2 := end of Popup-Group-List  
//          -1 := total end of Menu-List  
// text:    the name for the menu/popup-menu entry  
// fp:     Function to be activated on menu-selection
```

The dialog is described in the following AGSIDLGD structure:

```
#define AGSIDLGD struct AgsiDlgDat  
struct AgsiDlgDat {                    // every dialog has it's own structure  
  DWORD                iOpen;           // auto reopen dialog (pos := 'rc')  
  HWND                hw;             // Hwnd of Dialog  
  BOOL (CALLBACK *wp) (HWND hw, UINT msg, WPARAM wp, LPARAM lp);  
  RECT                rc;             // Position rectangle  
  void (*Update) (void);             // Update dialog content  
  void (*Kill) (AGSIDLGD *pM);       // Kill dialog  
  void                *vp;            // reserved for C++ Dialogs (Dlg *this)  
};
```

Input parameters: `pDym` Pointer to the AGSIMENU

Return value: TRUE if successful otherwise FALSE

Note: This function is allowed to call only during initialization process.

Example:

```
// Prototypes for forward references  
static void PeriUpdate (void);  
static void PeriKill    (AGSIDLGD *pM);  
static void PeriDisp    (AGSIMENU *pM);
```

```
// Peripheral Dialog
AGSIDLGD PeriDlg = { 0, NULL, NULL, { -1, -1, -1, -1, }, PeriUpdate, PeriKill };

// Peripheral Menu Item
AGSIMENU PeriMenu = { 1, "&A/D Converter" , PeriDisp, 0, IDD_ADCON, &PeriDlg };
Agsi.DefineMenuItem(&PeriMenu);
```

```
BOOL AgsiWriteSFR (AGSIADDR SFRAddress, DWORD dwValue,
                  DWORD dwMask);
```

This function is used to write a value with specified mask into the SFR. Written are only those bits which have at the same position the mask bit set.

Input parameters:

SFRAddress	Address of the SFR
dwValue	Value to write into the SFR
dwMask	Mask to use for writing

Return value: TRUE if successful otherwise FALSE

Examples:

```
AgsiWriteSFR(0xA8, 0x80, 0xFF); // Write 0x80 to the SFR at Address 0xA8
AgsiWriteSFR(0xA8, 0x80, 0x80); // Set the MSB bit of the SFR at Address 0xA8
// (other bits unchanged)
```

```
BOOL AgsiReadSFR (AGSIADDR SFRAddress, DWORD* pdwCurrentValue,
                  DWORD* pdwPreviousValue, DWORD dwMask);
```

This function is used to read the value from the SFR with specified mask. Read are only those bits which have at the same position the mask bit set (other bits are cleared).

Input parameters:

SFRAddress	Address of the SFR
pdwCurrentValue	Pointer to current Value of the SFR which will be read
pdwPreviousValue	Pointer to previous Value of the SFR which will be read
dwMask	Mask to use for reading

Return value: TRUE if successful otherwise FALSE

Example:

```
#define IE 0xA8
DWORD cIE, pIE;
AgsiReadSFR(IE, cIE, pIE, 0xFF);
```

```
BOOL AgsiSetSFRReadValue (DWORD dwValue);
```

This function is used to set the SFR read value (used in watch function with read access set for this SFR).

Input parameters:

dwValue	SFR Read Value
---------	----------------

Return value: TRUE if successful otherwise FALSE

Example:

```
Static void P1Read(void) { // function for read access of the SFR register P1
    ...
    AgsiSetSFRReadValue(P1 & PORT1); // P1 (Port1 SFR Value), PORT1 (Port1 pins)
}
```

---

**BOOL AgsiWriteVTR (AGSIVTR hVTR, DWORD dwValue);**

This function is used to write a value into the VTR.

Input parameters:

hVTR	VTR handle
dwValue	Value to write into the VTR

Return value: TRUE if successful otherwise FALSE

Examples:

```
DWORD port1;

union fv {          // float value union
    float f;
    DWORD DW;
} vref;

port1 = 0x01;
Agsi.WriteVTR(hPORT1, port1);    // Write port1 to VTR with hPORT1 handle (char VTR)

vref.f = 2.5;
Agsi.WriteVTR(hVREF, vref.DW);  // Write vref to VTR with hVREF handle (float VTR)
```

**BOOL AgsiReadVTR (AGSIVTR hVTR, DWORD\* pdwCurrentValue);**

This function is used to read the value from the VTR.

Input parameters:

hVTR	VTR handle
pdwCurrentValue	Pointer to current Value of the VTR which will be read

Return value: TRUE if successful otherwise FALSE

Examples:

```
DWORD port1;

union fv {          // float value union
    float f;
    DWORD DW;
} vref;

Agsi.ReadVTR(hPORT1, &port1);    // Read the VTR value with hPORT1 handle (char VTR) into port1
Agsi.ReadVTR(hVREF, &vref.DW);  // Read the VTR value with hVREF handle (float VTR) into vref
```

**BOOL AgsiWriteMemory (AGSIADDR Address, DWORD dwCount, BYTE\* pbValue);**

This function is used to write data into XDATA memory.

Input parameters:

Address	Start Address of Memory
dwCount	Number of bytes to write
pbValue	Pointer to buffer which data will be written

Return value: TRUE if successful otherwise FALSE

Example:

```
BYTE buffer[10];
AgsiWriteMemory(0x1000, 10, buffer);
```

---

```
BOOL AgsiReadMemory(AGSIADDR Address, DWORD dwCount,  
                    BYTE* pbValue);
```

This function is used to read data from XDATA memory.

Input parameters:

Address	Start Address of Memory
dwCount	Number of bytes to read
pBValue	Pointer to buffer in which data will be read

Return value: TRUE if successful otherwise FALSE

Example:

```
BYTE buffer[10];  
AgsiReadMemory(0x1000, 10, buffer);
```

```
BOOL AgsiSetTimer(AGSITIMER hTimer, DWORD dwClock);
```

This function is used to set the timer expire time in cycles.

Input parameters:

hTimer	Timer handle
dwClock	Number of machine cycles before the timer watch function is called

Return value: TRUE if successful otherwise FALSE

Example:

```
AgsiSetTimer(AdcCompleted, 10); // set timer (with AdcCompleted handle) to 10 cycles
```

```
UINT64 AgsiGetClock(void);
```

This function is used to get the number of cycles executed.

Return value: number of machine cycles executed

Example:

```
UINT64 cycles;  
Cycles = AgsiGetClock();
```

```
AGSIADDR AgsiGetProgramCounter(void);
```

This function is used to get the current program counter value (PC).

Return value: program counter value

Example:

```
AGSIADDR pc;  
pc = AgsiGetProgramCounter();
```

```
BOOL AgsiIsInInterrupt(void);
```

This function is used to examine if an interrupt is in progress.

Return value: TRUE if interrupt is in progress otherwise FALSE

Example:

```
if (AgsiIsInInterrupt()) { // interrupt in progress }  
else { // normal program execution }
```

---

**BOOL AgsiIsSleeping (void);**

This function is used to examine if the CPU sleep mode is active.

Return value: TRUE if CPU sleep mode is active otherwise FALSE

Example:

```
if (AgsiIsSleeping()) { // CPU is in sleep mode }  
else { // Normal CPU mode }
```

**void AgsiStopSimulator(void);**

This function is used to stop the simulation.

**void AgsiTriggerReset(void);**

This function is used to trigger a CPU reset.

**void AgsiUpdateWindows(void);**

This function is used to force  $\mu$ Vision2 to update all windows.