ADS
Applied Data Systems, Inc.

ADSmartIO Driver Specification for Windows CE

Version: 1.0

ADS document #110110-4004A

Last Saved: 12/21/00 2:51 PM

Applied Data Systems Inc.
9140 Guilford Road, Columbia, MD 21046
301-490-4007

Driver Change Log

Version	Release #	Date	By	Changes
Draft		11/10/2000	JB	Created
1.0	700110-4004B	12/08/2000	JB	Add SIOSetSSPTimeout()

Document Revision History

Revision	Date	By	Changes
	12/08/2000	JB	
A	12/21/00	AK	Initial release

Introduction

The ADSmartIO™ system enhances the I/O of ADS StrongARM products with configurable functionality and autonomous operation. ADSmartIO consists of an independent RISC microcontroller, firmware, filtering and protection circuitry and operating system drivers.

The ADSmartIO communicates with the StrongARM processor via an internal high-speed serial bus using an ADS protocol. Operating system drivers support this communication protocol and provide a library of functions with which to access the ADSmartIO functionality.

This manual describes the drivers for Windows CE.

1. Windows CE

Windows CE builds for ADS products display the ADSmartIO firmware version on the debug port during boot (eg. "`ATMEL 8535 Detected: Version: 0x4017`") Functionality of your ADSmartIO is dependent on the version built into your system.

The ADSmartIO driver for Windows CE consists of a DLL in the \Windows folder of your ADS StrongARM device and a corresponding library file required when developing your application. The files are keybddr.dll and keybdr.dll, respectively. SIOFunc.h is the C header file for the functions in the library.

Release 700110-4014B of the ADSmartIO library is compatible with the following ADS CE builds:

Windows CE 2.12

ADS Platform	ADS CE Version	Built-in Filename
GC Plus	Ver 2.22 and Later	Keybddr.dll
GC Master	Ver 2.19 and Later	Keybddr.dll
GC DUAL Master	Ver 2.19 and Later	Keybddr.dll
Bitsy	NA	NA

Windows CE 3.0

ADS Platform	ADS CE Version	Built-in Filename
GC Plus	Ver 3.01 and Later	Keybddr.dll
GC Master	NA	NA
GC DUAL Master	NA	NA
Bitsy	NA	NA

2. Functional Specification

This section describes the functions available to Windows CE developers using the ADSmartIO libraries.

void SIOSelectOption (BYTE Option)

Description: You can select one of three I/O options. Consult Section 3 for functionality.

Input: Option 1 ~ 3

Output: OK for success, ERROR if there is no room for the driver.

Version: Both 0x4017 and 0x8003

UINT SIOFirmwareVersion ()

Description: Read firmware version from ATMEL microprocessor.

Input: NONE

Output: return 16 bit integer value.

Version: Both 0x4017 and 0x8003

UINT SIOReadDeviceVersion ()

Description: read data indicating the ASCII character of the revision level of the SMART IO Micro code.

Input: NONE

Output: return 16 bit integer value.

Version: Both 0x4017 and 0x8003

UINT SIOReadDevicePartType ()

Description: read data indicating the ATMEL Micro-controller part type.

Input: NONE

Output: return 16 bit integer value.

Version: Both 0x4017 and 0x8003

For example, 0x8535 for ATMEL 8535 Chip

BOOL SIOSetKeyPadSize (BYTE x, BYTE y)

Description: Set the size of the keypad, 1≤x≤8, 1≤y≤8. The columns are driven using Port A, and the Rows are sensed using Port C. If a keypad matrix is less then 8x8, the unused pins can be used as digital/analog I/Os. The Port pins are used starting with bit 0. The Keypad scan will not start until Option 1 is set and SetKeypadSize are issued in that order. After Option1 is set Ports A and C will remain as inputs until a value is written with SetKeypadSize. To use Option 1 settings but disable keyscan, set row and column sizes to zero.
When a key is pressed, the AVR returns a number between 0 and 63 calculated as follows:

char = column + (row – 1) * 8

With row and column the coordinates of the key pressed (1 to 8). See the command SIOReadKBData() described below for reading the Keypad Data.

Input: X: the size of Rows
Y: the size of Columns

Output: 1 for success, 0 if there is error

Version: Both 0x4017 and 0x8003

BYTE SIOReadKeypad ()

Description: Read keypad Data

Input: NONE

Output: Returns keypad Data, 0xff if there's timeout

Version: Both 0x4017 and 0x8003

Keypad data returned depends on firmware version:
Version 0x4017 scans a matrix keypad up to an 8x8 keys in size. It returns a value from 0 to 63 when a key is depressed. No value is returned when a key is released.
Version 0x8003 is designed for a 10-switch keypad (10-pole, single-throw). The firmware connects internal pullups to each channel. Signals ROW0 to ROW7 and COL0 to COL1 constitute the ten channels. They return 0x30 to 0x39 when a key is pressed (connected to ground), 0xB0 to 0xB9 when a key is released.

UINT SIOAnalogConversion (BYTE channel)

Description: Read A/D Data from specified channel.
Input: Channel Number
Output: 16 bit Value with 10 Bit A/D Data, 0xffff if there's timeout.
Version: Both 0x4017 and 0x8003

BOOL SIOSetSpeaker (BYTE Duration, BYTE Tone)

Description: set the speaker with duration and tone.
Input: Duration is the hexadecimal value of the duration of the tone
Tone is a hexadecimal value proportional to the period (1/frequency) of the tone
Output: 16 bit A/D data, 0xffff if there's timeout.
Version: Only 0x8003

Duration (sec) = (Duration + 1) * 0.05 sec:
Duration = 0x0 0.05 sec duration
Duration = 0xFF12.8 sec duration

Frequency (Hz) = 1/[(Tone + 1) * 34.72 usec]:
Tone = 0x1 2 * 34.72 usec period (= 14.4 kHz)
Tone = 0xFE 255 * 34.72 usec period (= 112.95 Hz)
Tone = 0xFF 0 Hz

UINT SIOReadVoltage ()

Description: Read Voltage Level
Input: NONE
Output: 16 bit Voltage Level
Version: Only 0x8003

UINT SIOReadTemperature ()

Description: Read Temperature Level
Input: NONE
Output: 16 bit Voltage Level
Version: Only 0x8003

void SIOBackLightPwm(BYTE value)

Description: Set Back Light Level
Input: 0xff for Low Intensity, 0 for Maximum Intensity
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOVEEPwm(BYTE value)

Description: Set Contrast Level
Input: 0 for Low Contrast, 0xff for Maximum Contrast
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOBackLightON()

Description: Turn on the BackLight
Input: NONE
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOBackLightOFF()

Description: Turn off the BackLight
Input: NONE
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOWritePortA (BYTE value)

Description: Write data to Port A
Input: byte data for writing
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOWritePortB (BYTE value)

Description: Write data to Port B
Input: byte data for writing
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOWritePortC (BYTE value)

Description: Write data to Port C
Input: byte data for writing
Output: NONE
Version: Both 0x4017 and 0x8003

void SIOWritePortD (BYTE value)

Description: Write data to Port D
Input: byte data for writing
Output: NONE
Version: Both 0x4017 and 0x8003

UINT SIOReadPortA ()

Description: Read data from Port A
Input: NONE
Output: 16 Bit Value with 8 bit data
Version: Both 0x4017 and 0x8003

UINT SIOReadPortB ()

Description: Read data from Port B
Input: NONE
Output: 16 Bit Value with 8 bit data
Version: Both 0x4017 and 0x8003

UINT SIOReadPortC ()

Description: Read data from Port C
Input: NONE
Output: 16 Bit Value with 8 bit data
Version: Both 0x4017 and 0x8003

UINT SIOReadPortD ()

Description: Read data from Port D
Input: NONE
Output: 16 Bit Value with 8 bit data
Version: Both 0x4017 and 0x8003

void SIOInitDDRA (BYTE value)

Description: Set the data direction for Port A

Input: Data Direction Value

Output: NONE

Version: Both 0x4017 and 0x8003

DDRA Register:

A7	A6	A5	A4	A3	A2	A1	A0

Where:
0 = Configured as input
1 = Configured as output

void SIOInitDDRB (BYTE value)

Description: Set the data direction for Port B

Input: Data Direction Value

Output: NONE

Version: Both 0x4017 and 0x8003

DDRB Register:

NA	NA	NA	NA	B3	B2	B1	B0

Where:
0 = Configured as input
1 = Configured as output

void SIOInitDDRC (BYTE value)

Description: Set the data direction for Port C

Input: Data Direction Value

Output: NONE

Version: Both 0x4017 and 0x8003

DDRC Register:

C7	C6	C5	C4	C3	C2	C1	C0

Where:
0 = Configured as input
1 = Configured as output

void SIOInitDDRD (BYTE value)

Description: Set the data direction for Port D

Input: Data Direction Value

Output: NONE

Version: Both 0x4017 and 0x8003

DDRD Register:

NA	NA	NA	NA	NA	NA	D1	D0

Where: 0 = Configured as input
1 = Configured as output

BOOL SIOSetSSPTimeout (ULONG Timeout)

Description: Set Timeout for waiting response from SMARTIO Chip

Input: Timeout (unit: msec)

Output: Return TRUE

Version: Both 0x4017 and 0x8003

ADS CE version 2.22 and later on CE 2.12

ADS CE version 3.03 and later on CE 3.0

3. Smart IO configurations

The following are the I/O configurations supported by the ADSmartIO firmware. Use the SIOSelectOption() function to select the option required by your application.

	Option 1	Option 2	Option 3	GC Plus pin number	
	8x 8 Keypad + 7 Digital IOs	**8 Analog Inputs + 15 Digital IOs**	**23 Digital Ios**	**J2**	**J7**
PA0	Keypad Column 0	Analog Input 0	Digital IO A0		17
PA1	Keypad Column 1	Analog Input 1	Digital IO A1		19
PA2	Keypad Column 2	Analog Input 2	Digital IO A2		21
PA3	Keypad Column 3	Analog Input 3	Digital IO A3		23
PA4	Keypad Column 4	Analog Input 4	Digital IO A4		25
PA5	Keypad Column 5	Analog Input 5	Digital IO A5		27
PA6	Keypad Column 6	Analog Input 6	Digital IO A6		29
PA7	Keypad Column 7	Analog Input 7	Digital IO A7		31
PB0	Digital IO B0	Digital IO B0	Digital IO B0	10	
PB1	Digital IO B1	Digital IO B1	Digital IO B1	12	
PB2	Digital IO B2	Digital IO B2	Digital IO B2	14	
PB3	Digital IO B3	Digital IO B3	Digital IO B3	16	
PB4	Digital IO B4	(/SS)	Digital IO B4	18	
PC0	Keypad Row 7	Digital IO C0	Digital IO C0		15
PC1	Keypad Row 6	Digital IO C1	Digital IO C1		13
PC2	Keypad Row 5	Digital IO C2	Digital IO C2		11
PC3	Keypad Row 4	Digital IO C3	Digital IO C3		9
PC4	Keypad Row 3	Digital IO C4	Digital IO C4		7
PC5	Keypad Row 2	Digital IO C5	Digital IO C5		5
PC6	Keypad Row 1	Digital IO C6	Digital IO C6		3
PC7	Keypad Row 0	Digital IO C7	Digital IO C7		1
PD0	Digital IO D0	Digital IO D0	Digital IO D0	20	
PD1	Digital IO D1	Digital IO D1	Digital IO D1	22	
PD2	PS2 Clock	PS2 Clock	PS2 Clock		
PD3	*Reserved*	*Reserved*	*Reserved*		
PD4	PWM VEE	PWM VEE	PWM VEE		
PD5	PWM Backlight	PWM Backlight	PWM Backlight		
PD6	PS2 Data	PS2 Data	PS2 Data		
PD7	Backlight ON/OFF	Backlight ON/OFF	Backlight ON/OFF		

4. Sample Code

```
#include <windows.h>
#include "SIOFunc.h"
#define DIS_COUNT 10

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPTSTR lpCmdLine, int nCmdShow )
{
        int Count=0, i;
        BYTE data=0xff;
        BOOL bRet;
        UINT DeviceVer ,Duration, Tone, fwVersion, Voltage, Temperature;

        RETAILMSG(1,(L" TestAvr: Start...\r\n"));

        fwVersion = SIOFirmwareVersion();
        RETAILMSG(1,(L" TestAvr: Firmware Version = 0x%x...\r\n",fwVersion));

        DeviceVer = SIOReadDeviceVersion();
        RETAILMSG(1,(L" TestAvr: Device Version = 0x%x...\r\n",DeviceVer));

        SIOBackLightOFF();
        Sleep(1000);
        SIOBackLightON();

        for(i=8;i > 0;i--)
        {
                SIOSelectOption(1);// select Option1
                RETAILMSG(1,(L"\r\n TestAvr : SIOSelectOption(1)\r\n"));

                bRet = SIOSetKeyPadSize(i,i);
                RETAILMSG(1,(L"\r\n TestAvr : Set KeypadSize (%d,%d). \r\n",i,i));
                if(!bRet)
                {
                        RETAILMSG(1,(L"\r\n TestAvr : Error in Set KeypadSize (%d,%d). \r\n",i,i));

                        continue;
                }
                else
                {
                        Count=0;
                        do
                        {
                                data = SIOReadKeypad();

                                if((data & 0x30) == 0x30)
                                {
                                        RETAILMSG(1,(L"TestAvr : Pressed = %d \r\n",data));

                                }
                                else if((data & 0xB0) == 0xB0)
                                {
                                        RETAILMSG(1,(L"TestAvr : Released = %d \r\n",data));

                                }
                                else if(data == 0xff)
                                {
                                        RETAILMSG(0,(L"TestAvr : Time out !\r\n"));
                                }
                                Sleep(100); // 100 msec delay
                                Count++;
                        }while(Count < DIS_COUNT);
                 }
        }
```

```
                    RETAILMSG(1,(L" TestAvr: Speaker \r\n"));
                    Duration =16;
                    Tone = 16;
                     for(i=0;i < 5  ;i++)
                    {
                         SIOSetSpeaker(Duration,Tone);
                         RETAILMSG(1,(L" TestAvr: Speaker = (%d, %d) \r\n",i,i));
                        Sleep(1000);
                        Tone *= 2;
                     }

                     for(i=1;i<=10 ;i++)
                    {
                           Voltage = SIOReadVoltage();
                           RETAILMSG(1,(L" TestAvr: Voltage = %d \r\n",Voltage));
                           Sleep(1000);
                     }

                     for(i=1;i <= 10 ;i++)
                     {
                            Temperature = SIOReadTemperature();
                            RETAILMSG(1,(L" TestAvr: Temperature = %d \r\n",Temperature));
                             Sleep(1000);
                     }

                     RETAILMSG(1,(L" TestAvr: Done...\r\n"));

                      return 0;
}
```