Application Note

AN2263/D
12/2002

PC Master Software: Creation of Advanced Control Pages

By Milan Brejl
S^{3}L Applications Engineering
Motorola Czech Systems Laboratories
Roznov pod Radhostem, Czech Republic

Objectives

This application note describes the advanced techniques for creating PC master software Control Pages. For example, it can be represented by the design of a framed Control Page composed of two frames, with an HTML-based frame and Microsoft® Excel running embedded in the second frame. This enables you to control and adjust the target board application parameters and to visualize and analyze the board application processes. An adaptive signal-prediction algorithm running on the target board is used as a comprehensive example.

Prerequisites

This application note assumes basic knowledge of:

- PC Master
- Creation of HTML pages
- Basic HTML scripting with VB Script
- Creation of Microsoft Excel applications

Introduction

PC master software is a PC Windows®-based application providing various support for embedded applications. It is a tool that can be used for debugging, monitoring, and controlling the target application and also for demonstrating the target board application functionality.

Microsoft and Windows are registered trademarks of Microsoft Corporation in the U.S. and/or other countries.

The embedded application runs on a target board (see **Figure 1**) which is connected to the PC via an RS232 interface. The PC runs the PC master software application that communicates with the target board application. The PC master software can read and write the target board application variables, record variable value courses, and provides other functions useful for monitoring, controlling, or debugging the target board application. For more information about this tool, refer to the *PC Master Software User Manual* portion of the *MSW3SDK000AA: Embedded Software Development Kit for 56800/56800E* and to the application note *PC Master Software Usage* (Motorola document order number AN2395/D). Both of these documents are available on the World Wide Web at:

http://motorola.com/semiconductors/

The Control Page is one of several pages which can be displayed in the *Detail view* pane of the PC master software main application window. It is intended to enable control of the target board application and may also contain parts dedicated to monitoring or analysis of the processes running on the target board. The page is based on HTML which is widely used on the Internet so the design can be made with the help of the web-authoring tools which are commercially available or even free on the Internet.

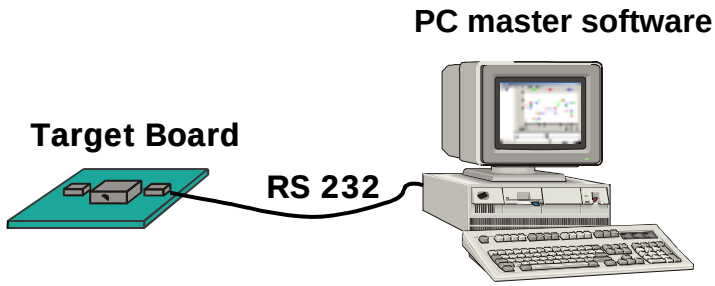


Figure 1. PC Master Software and Target Board Application

Control Page Functionality

PC master software uses Microsoft Internet Explorer to display the HTML pages, which enables the user to use a wide range of web technologies supported by it. Internet Explorer allows the following technologies to be used from the PC master software environment:

- HTML
- Dynamic HTML (DHTML)
- Java™Script or VBScript
- ActiveX

Java and Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and other countries.

PC master software application implements an ActiveX object, which is used to enable access to and control of the target board application. The functions exposed by the ActiveX object can be called from a script inserted in the HTML code or from other Windows applications that are capable of working with ActiveX objects. In this document it will be demonstrated how to access the PC master software functions from both VB Script and Microsoft Excel running embedded in PC master software.

The PC master software ActiveX control provides the following functions:

- **ReadVariable** reads a value from a PC master software-defined variable
- **WriteVariable** writes a value to a PC master software-defined variable
- **SendCommand** sends a PC master software-defined command
- **SendCommandDlg** opens a command dialog and sends a PC master software-defined command

PC master software version 1.1 adds the following functions:

- **ReadMemory, ReadMemoryHex** reads a block of memory from a specified location
- **GetCurrentRecorderData** retrieves data currently displayed in the recorder chart
- **GetCurrentRecorderSerie** retrieves one data series from the currently-displayed recorder chart
- **StartCurrentRecorder** starts the currently-displayed recorder
- **StopCurrentRecorder** stops the currently-displayed recorder
- **GetCurrentRecorderState** retrieves the current recorder status code and status text of the currently-displayed recorder

PC master software version 1.1 also implements one call back event:

- **OnRecorderDone**, called when the currently-selected recorder finishes downloading new recorder data

See the *PC Master Software User Manual* for a detailed description of each function.

Example: Step-by-Step Guide

The rest of this document gives a step-by-step example of the creation of a complex PC master software Control Page. The following text describes:

- The target board application
- Application variables that are used in the interface between PC master software and the board application
- The control and monitor techniques used on the Control Page
- The actual design of the Control Page

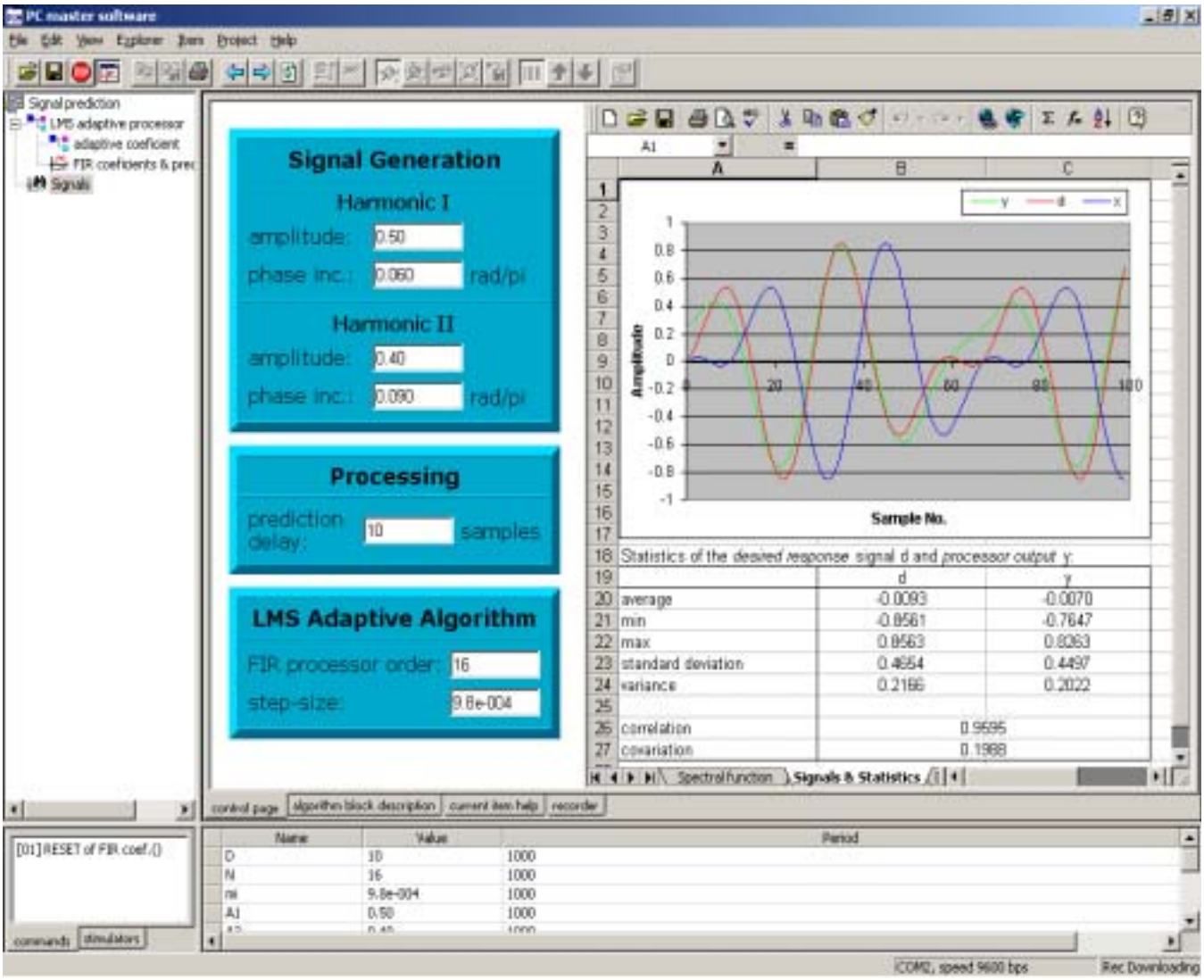


Figure 2. PC Master Software with Control Page Consisting of an HTML Frame and an Excel Frame

Target Board Application Description

The target board application block diagram is shown in **Figure 3**.

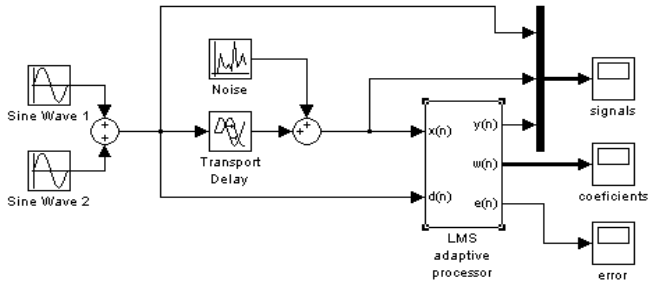


Figure 3. Board Application Block Diagram

The source signal is generated by the application as the sum of two sine waves with adjustable amplitudes and frequencies (phase increments per sample). The signal is delayed by an adjustable number of samples (*D*) and this delayed signal is connected to the input *x*(*n*) of the LMS (least mean square) adaptive processor. The original version of the source signal is connected to the processor desired response *d*(*n*). The aim of the adaptive processor is to minimize the error signal *e*(*n*) that is the difference between the output signal *y*(*n*) — produced by a FIR (finite input response) system from the input *x*(*n*) — and desired response *d*(*n*). As the result, the adaptive processor builds a model of the incoming signal such that it can predict *D* samples of source signal. The order (*N*) and the adaptation step-size (*mi*) of the adaptive FIR processor is also adjustable in the application.

All the target board application parameters can be adjusted using the watch window in the PC master software application. The signals generated in the target board application can be monitored in the oscilloscope and recorder charts. **Figure 4** shows the PC master software recorder chart displaying the following application signals:

- Processor input signal *x*(*n*)
- Processor desired response *d*(*n*)
- Processor output signal *y*(*n*)

The more the processor is adapted, the closer the output signal is to the desired response.

The PC master software oscilloscope in **Figure 5** shows the evolution of the processor FIR coefficients **w**(*n*) during the adaptation together with the error signal *e*(*n*).

Appendix A contains the C-language source code of the target board application main routine.

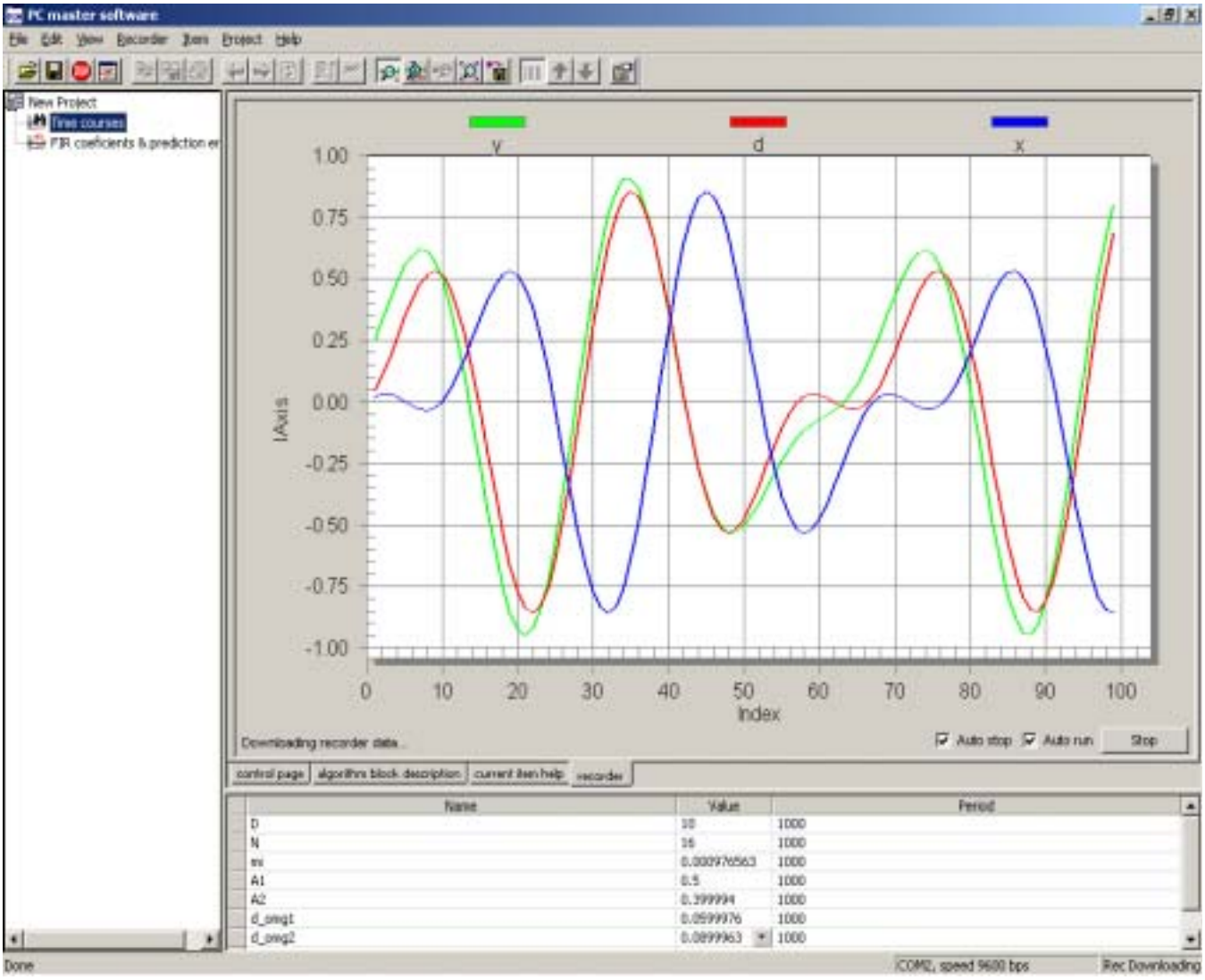


Figure 4. PC Master Software Recorder with Signals *x*(*n*), *y*(*n*) and *d*(*n*) and Some Application Parameters in PC Master Software Watch

Control Page Proposal

Our goal is to create a Control Page composed of an HTML frame and an Excel frame. The Control Page will:

1. **Provide an interface to monitor and adjust all application parameters, making a more user-friendly environment than the PC master software watch**
   - A graphical Control Panel will be created in HTML and will be enhanced by scripts to provide control functionality.

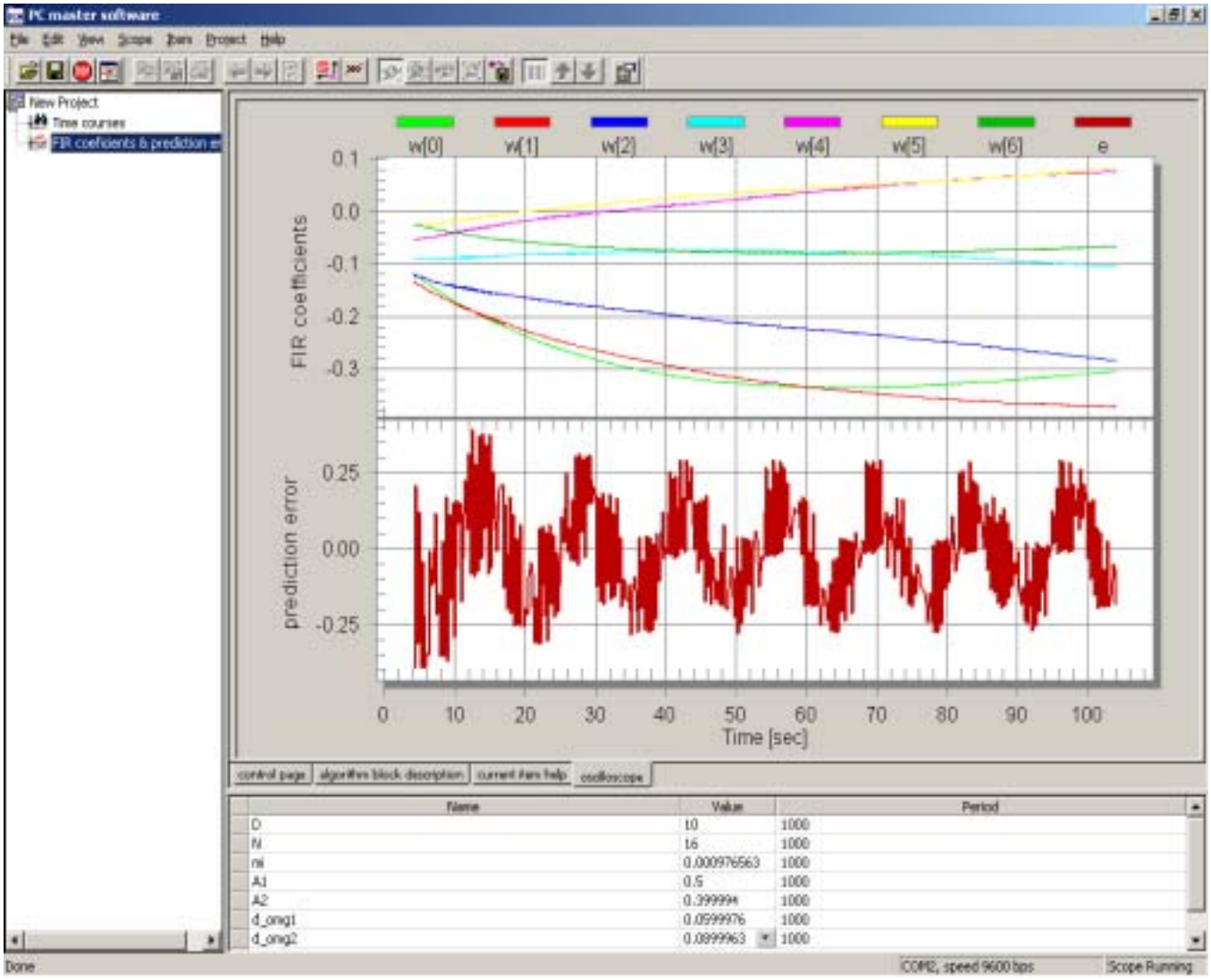


Figure 5. Oscilloscope Displaying the Time Course of FIR Coefficients *w*(*n*), signal *e*(*n*), and Some of the Application Parameters in PC Master Software Watch

1. **Analyze the signal model and get a Spectral Function calculated from FIR coefficients w(*n*)**
   - The calculation and visualization of the Spectral Function will be done in Microsoft Excel.
   - It will be described how to use Microsoft Excel embedded in the Control Page and how to access the **w**(*n*) coefficients in Excel Visual Basic script.

2. **Calculate the statistics of the signals captured by the PC master software recorder**
   - Microsoft Excel functions like MIN, MAX, AVERAGE, STDEV, VAR, and CORREL will be used for this task.
   - It will be described how to load the PC master software recorder signal into Microsoft Excel cells for further processing.

Control Panel Creation

The Control Panel should be designed to allow the adjustment of all application parameters (see **Figure 6**).

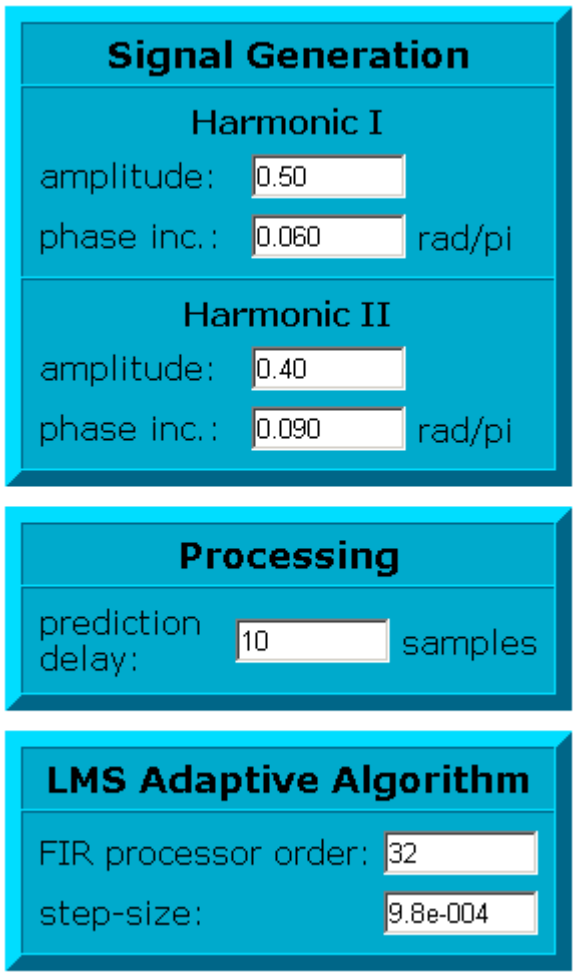


Figure 6. HTML Control Panel

To start the creation of the Control Page, first prepare the HTML source of the three control tables, as displayed in **Figure 6**. The HTML "TABLE" tag with color attributes can be used to achieve the 3D look. The edit boxes can be implemented with the "INPUT" tag. The source code of the first board named *Signal Generation* might look like this:

```
<table bgcolor="#00aacc" border="8" bordercolordark="#006688"
       bordercolorlight="#00ddff" cellpadding="4"
       cellspacing="0" width="100%">
  <tr>
    <td align="center" valign="middle">
      <font size="4"><b>Signal Generation</b></font>
    </td>
  </tr>
  <tr>
    <td>
      <table border="0" cellpadding="2" cellspacing="2"
             width="100%">
        <tr>
          <td colspan="2" align="center">
            <b>Harmonic I</b>
          </td>
        </tr>
        <tr>
          <td>amplitude:</td>
          <td>
            <input type="text" name="A1" size="7">
          </td>
        </tr>
        <tr>
          <td nowrap>phase inc.:</td>
          <td nowrap>
            <input type="text" name="d_omg1" size="7"> rad/pi
          </td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td>
      <table border="0" cellpadding="2" cellspacing="2"
             width="100%">
        <tr>
          <td colspan="2" align="center">
            <b>Harmonic II</b>
          </td>
        </tr>
        <tr>
          <td>amplitude:</td>
          <td>
            <input type="text" name="A2" size="7">
          </td>
```

```
          </tr>
          <tr>
             <td nowrap>phase inc.:</td>
             <td nowrap>
                <input type="text" name="d_omg2" size="7"> rad/pi
             </td>
          </tr>
       </table>
    </td>
  </tr>
</table>
```

The input boxes from the first table display the following parameters:

- Amplitude of Harmonic I — input box named "A1"
- Phase increment of Harmonic I — input box named "d_omg1"
- Amplitude of Harmonic II — input box named "A2"
- Phase increment of Harmonic II — input box named "d_omg2"

The input boxes are named the same as the corresponding PC master software variables. This establishes an assignment between the input box and the PC master software variable that will be used when writing a new value from the input box into the board application.

The other two tables can be designed using the first table as an example. The parameters displayed in these tables are (see **Figure 6**):

- Prediction delay — input box named "D"
- FIR processor order — input box named "N"
- LMS processor step-size — input box named "mi"

Refer to **Appendix B** for the full HTML source code of these two boards.

Scripting

To enable the HTML page to communicate with PC master software functions, insert the PC master software ActiveX control at the beginning of the HTML *body* section. The control is identified by the unique global Class ID number (see the example below). Since the object itself does not provide a graphical interface, it is possible to set the object dimensions (height and width) to zero to make it invisible.

```
<object name="pcm" width="0" height="0"
classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153"></object>
```

The HTML page created above is enhanced by scripting which is implemented by using the following HTML code. VBScript was selected for this example.

When the page is loaded the values of all input boxes are filled by the appropriate parameter values from the target board application. Create the *ReadSettings* sub-routine to perform this task and let the sub-routine be called after the Control Panel is loaded. Add a call to this procedure at the end of the HTML *body* section.

```
  <script language="VBScript">
    ReadSettings
  </script>
```

The sub-routine body is like this:

```
<script language="VBScript">

Sub ReadSettings()

  Dim succ,v,tv,retMsg
```

Use the *ReadVariable* function of the PC master software ActiveX object in order to fill the first input box (Harmonics I amplitude in the example).

```
  succ = pcm.ReadVariable("A1",v,tv,retMsg)
```

This function gets as the first parameter the name of the PC master software variable. The next two variables passed to the call are filled the variable value (*v*) and the value in text format (*tv*) according to the setting in PC master software. The last parameter (*retMsg*) is filled by an error message in case there is an error. The function call returns *true* (and *v* and *tv* are filled) if the function was performed successfully. It is *false* if it fails for any reason (and *retMsg* is filled).

```
  If succ then
    A1.Value = tv
  else
    A1.Value = retMsg
  End If
```

Repeat this part for all application parameters to fill all control panel input boxes: d_omg1, A2, d_omg2, D, N, mi (see **Appendix B**).

```
End Sub

</script>
```

To enable the manual adjustment of the parameters displayed in the input boxes, equip each input box with two event procedures. The event calls a sub-routine and the sub-routine calls the target board via PC master software activeX object.

```
<input type="text" name="A1" size="7" onchange="WriteValue"
onkeydown="WriteValueOnEnter">
```

The *onchange* event is automatically invoked by the Internet Explorer when the cursor leaves the edit box. This event is handled the same way in all input boxes by the WriteValue sub-routine. Here, the *SrcElement* property of the *Event* object is used to determine which edit box to handle.

Use the *WriteVariable* function of the PC master software ActiveX object to write the adjusted value.

```
Sub WriteValue

  Dim succ,retMsg

  succ = pcm.WriteVariable(Window.Event.SrcElement.Name,
                           Window.Event.SrcElement.Value,retMsg)
```

This function gets the name of the PC master software variable as the first parameter. The name of the input box is the same as the name of the variable. The next parameter is the new value to be written. As in the other PC master software ActiveX functions, the return value can be used to get the success of the call.

```
  If succ then
    txtError.InnerText = ""
  else
    txtError.InnerText = retMsg
  End If

End Sub
```

As the user might expect, the new value should be written to the board application when he or she presses Enter. That is why the *onkeydown* event is also handled in all input boxes. The handling sub-routine *WriteValueOnEnter* tests the pressed key and if it is the Enter key the value is written with the help of the *WriteValue* sub-routine.

```
Sub WriteValueOnEnter

  If chr(Window.Event.KeyCode) = VBCR Then
    WriteValue
  End If

End Sub
```

Appendix B contains the complete source code of the Control Panel.

Framed Control Page

The Control Page consists of two frames (see **Figure 7**). The left one is the previously created Control Panel (file *control_panel.html*) and the right one is composed by the Microsoft Excel file (*LMS.xls*) which is described in the following text.

The HTML file that creates the placement of both frames is the actual main Control Page source and looks like this:

```
<html>

  <head>
    <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1">
    <meta name="generator" content="Adobe GoLive 4">
    <title>Control Page</title>
  </head>

  <frameset cols="320,*" framespacing="0" border="4"
   frameborder="YES" bordercolor="#dddddd">
    <frame src="control_panel.html" name="HTML">
    <frame src="LMS.xls" name="Excel">
  </frameset>
</html>
```

Internet Explorer automatically runs the Microsoft Excel application embedded in the frame to display the contents of the .xls file.

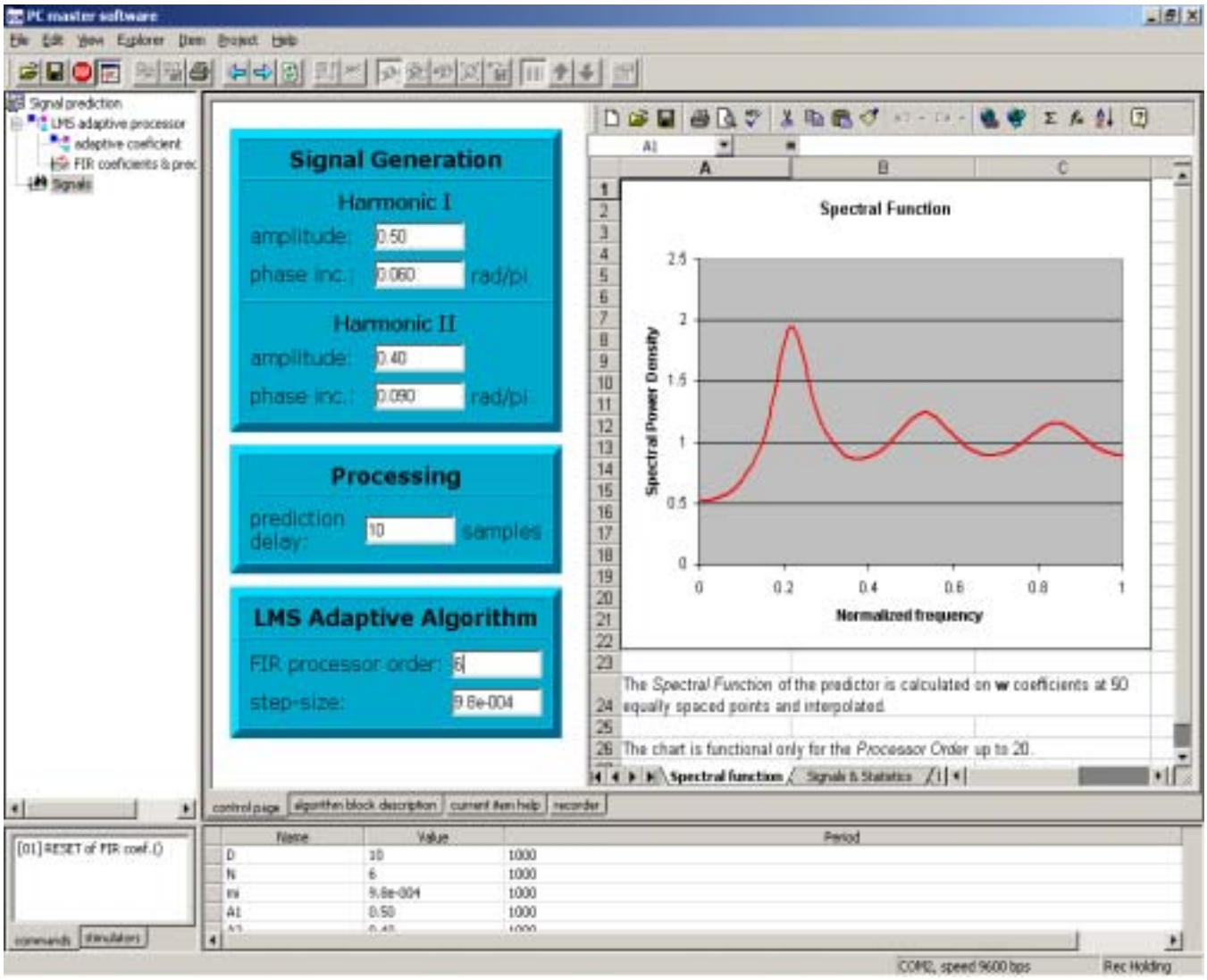


Figure 7. PC Master Software with Control Page and Spectral Function Sheet Displayed in the Excel Frame

Microsoft Excel Data Analysis

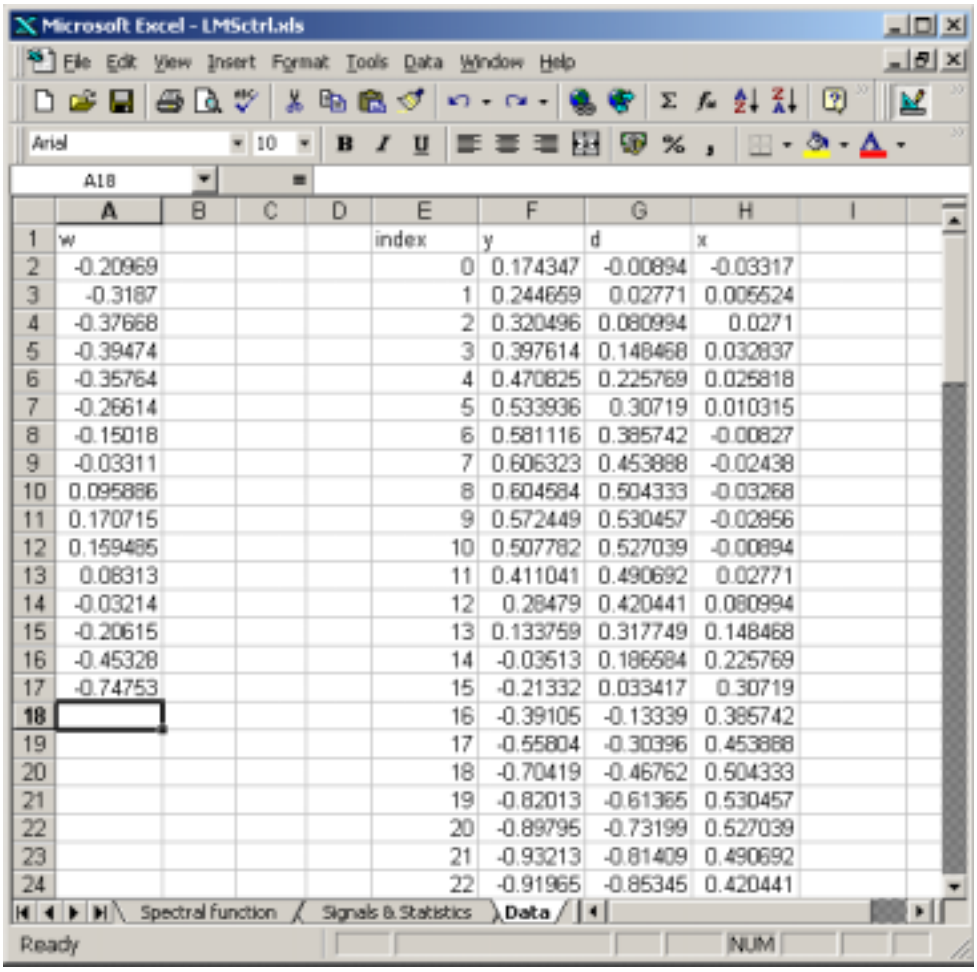
Microsoft Excel workbook embedded in the second frame of the Control Page will have three worksheets:

- The first one will be used to display the frequency function (see **Figure 7**) of the signal model calculated from the FIR coefficients **w**(*n*). The number of **w**(*n*) coefficients is equal to the order *N* which is one of the application parameters.
- The second sheet will display the recorded signals *x*(*n*), *y*(*n*) and *d*(*n*) and some statistics calculated from the signals (see **Figure 2**).
- The third sheet serves as the data interface to the PC master software.

It is out of the scope of this document to describe the creation of the first two Excel worksheets. Refer to the *Excel User Manual* for the implementation details.

The data are imported into the third worksheet (see **Figure 8**) using Excel Visual Basic script. Microsoft Excel is also capable of running ActiveX-aware scripts. That is why the connection from Excel to PC master software can be accomplished through the same PC master software functions as listed in **Control Page Functionality**.

Column A of the third worksheet holds the *N* FIR coefficients **w**(*n*). Here can be from 1 to 128 coefficients so the sheet range assigned to **w**(*n*) coefficients is "A2:A129". The columns "E:H" hold the time index and the signals *y*(*n*), *d*(*n*) and *x*(*n*) (see **Figure 8**). The following description explains how to fill these cells and columns with the PC master software data.



	A	E	F	G	H
1	w	index	y	d	x
2	-0.20969	0	0.174347	-0.00894	-0.03317
3	-0.3187	1	0.244659	0.02771	0.005524
4	-0.37668	2	0.320496	0.080994	0.0271
5	-0.39474	3	0.397614	0.148468	0.032837
6	-0.35764	4	0.470825	0.225769	0.025818
7	-0.26614	5	0.533936	0.30719	0.010315
8	-0.15018	6	0.581116	0.385742	-0.00827
9	-0.03311	7	0.606323	0.453888	-0.02438
10	0.095886	8	0.604584	0.504333	-0.03268
11	0.170715	9	0.572449	0.530457	-0.02856
12	0.159485	10	0.507782	0.527039	-0.00894
13	0.08313	11	0.411041	0.490692	0.02771
14	-0.03214	12	0.28479	0.420441	0.080994
15	-0.20615	13	0.133759	0.317749	0.148468
16	-0.45328	14	-0.03513	0.186584	0.225769
17	-0.74753	15	-0.21332	0.033417	0.30719
18		16	-0.39105	-0.13339	0.385742
19		17	-0.55804	-0.30396	0.453888
20		18	-0.70419	-0.46762	0.504333
21		19	-0.82013	-0.61365	0.530457
22		20	-0.89795	-0.73199	0.527039
23		21	-0.93213	-0.81409	0.490692
24		22	-0.91965	-0.85345	0.420441

Figure 8. The Third ("*Data*") Worksheet in the Microsoft Excel Application

PC Master Software ActiveX control in Visual Basic

The PC master software data can be accessed in Microsoft Excel using a Visual Basic code that calls the PC master software ActiveX control functions.

Similar to the HTML based Control Panel, the PC master software ActiveX object must be created first:

```
Public pcm As McbPcm

Set pcm = CreateObject("MCB.PCM")
```

The ActiveX functions can now be called. For example, to load an array of w(*n*) coefficients use the *ReadMemory* function of the ActiveX object.

```
    Dim succ As Boolean

    succ = pcm.ReadMemory("w",16,data,msg)
```

The address of the memory block to be read is the first parameter of the *ReadMemory* function. This can be either an absolute numeric address or a symbol name valid in the current PC master software project. The second parameter is the size of the memory block to be read in bytes. The memory data are returned to the variable that is the third parameter. The last parameter is the variable for an error message in case of an error.

Usually, it is necessary to transform the data bytes into another data representation. For example, into an array of 16-bit fractional values — get the pairs of MSB (most significant) and LSB (least significant) bytes from the read memory block and put these bytes together into a fractional value:

```
    If succ Then

        For s = 0 To 7
            MSB = data(2 * s + 1)
            LSB = data(2 * s)
            If MSB >= 128 Then
                fracValue = (256 * MSB + LSB - 65535) / 32768
            Else
                fracValue = (256 * MSB + LSB) / 32768
            End If
        Next s

    Else
        MsgBox (msg)
    End If
```

Use the function *GetCurrentRecorderData* to load the PC master software recorder series. The function returns the matrix (2-dimensional array) of the recorder data in the first parameter variable (*data*), the array of series names in the second (*serNames*), time base (if defined) in the third (*tbase*) and the error message is the last parameter variable (*msg*).

```
succ = pcm.GetCurrentRecorderData(data,serNames,tbase,msg)
```

To get the bounds of the recorder data matrix use the *UBound* function.

```
If succ Then

    serCnt = UBound(data, 1)
    valCnt = UBound(data, 2)

Else
    MsgBox (msg)
End If
```

To put a data into the Excel sheet, first create a *SheetData* variable. Then use the *ClearContents* function of the sheet range object to clear the specified range contents. Assign the cells *Value* property to place the data value into the specified cell.

```
Dim sht As SheetData

Set sht = Worksheets("Data")

sht.Range("A1:B8").ClearContents
sht.Cells(1,2).Value = someValue
```

Back to the Example

Use these examples in our workbook now. Open the Visual Basic Editor in Microsoft Excel. Open the code editor of the *ThisWorkbook* object. Create a new module *Module1* and open it as well. Create a global declaration of the PC master software ActiveX object variable *pcm* and *ScheduleTime* in *Module1*.

```
Public pcm As McbPcm
Public ScheduleTime As Variant
```

Create an event-procedure in *ThisWorkbook* that will be activated when the Excel file is opened:

```
Private Sub Workbook_Open()

    Set pcm = CreateObject("MCB.PCM")

    ScheduleTime = Now + TimeValue("00:00:05")
    Application.OnTime ScheduleTime, "RecordAndAnalyze"

End Sub
```

This sub-routine assigns the PC master software ActiveX object to the *pcm* variable and schedules the sub-routine *RecordAndAnalyze* to run 5 seconds later.

Create the sub-routine *RecordAndAnalyze* in *Module1*. The purpose of this routine is to get the FIR coefficient values **w**(*n*) for the spectral function calculation and to get the recorder signals *x*(*n*), *y*(*n*) and *d*(*n*) for their statistics. The data will be saved to the specified positions on the third sheet named "Data". Then, the Excel cell functions will use this data to automatically calculate the Spectral Function and the signal statistics.

Create the declarations and assign a sheet variable with the third sheet named "Data" in the sub-routine *RecordAndAnalyze*.

```
Sub RecordAndAnalyze()

    Dim succ As Boolean
    Dim sht As SheetData

    Set sht = Worksheets("Data")
```

To get the **w**(*n*) FIR coefficients the PC master software ActiveX control function *ReadMemory* will be used because the **w**(*n*) coefficients are stored in an array in the board application memory. It is necessary to know the length of this array which corresponds to the processor order *N*.

```
    succ = pcm.ReadVariable("N", N, tv, msg)

    If succ Then
```

When the *N* variable is read successfully, 2**N* bytes of the **w**(*n*) array can be read into variable *data*.

```
        succ = pcm.ReadMemory("w", 2 * N, data, msg)

        If succ Then
```

When the *data* are read successfully, first clear the contents of column A rows 2 to 129 on the sheet "Data".

```
            sht.Range("A2:A129").ClearContents
```

Check the bounds of the read *data*

```
            serCnt = UBound(data)
```

and transform the *data* from the 8-bit memory dump into the 16-bits signed fractional values and write them to the cleared cells in order.

```
            For s = 0 To (serCnt \ 2)
                MSB = data(2 * s + 1)
                LSB = data(2 * s)
                If MSB >= 128 Then
                   sht.Cells(s + 2, 1).Value = ...
                             (256 * MSB + LSB - 65535) / 32768
                Else
                    sht.Cells(s + 2, 1).Value = ...
                                  (256 * MSB + LSB) / 32768
                End If
            Next s

        Else
            MsgBox (msg)
        End If

    Else
        MsgBox (msg)
    End If
```

The previous part of the sub-routine gets the data for the spectral function calculation. The next part loads the recorder signals for the signal statistics. The PC master software ActiveX component function *GetCurrentRecorderData* will be used in its variant for the VisualBasic *GetCurrentRecorderData_t* that is faster because it uses a typed array instead of variant array (see the *PC Master Software User Manual*).

```
succ = pcm.GetCurrentRecorderData_t(data,serNames,tbase,msg)
```

When the *data* are read successfully, clear the contents of columns E to H (column E for the time axis and the next three columns for signals) on the sheet "Data".

```
If succ Then

    sht.Range("E:H").ClearContents
```

Check the bounds of the read *data*.

```
    serCnt = UBound(data, 1)
    valCnt = UBound(data, 2)
```

Check if the read time base is specified

```
    If tbase > 0 Then
        sht.Cells(1, 5).Value = "time [sec]"
    Else
        sht.Cells(1, 5).Value = "index"
        tbase = 1
    End If
```

and fill the X-axis column.

```
    For i = 0 To valCnt
        sht.Cells(i + 2, 5).Value = i * tbase
    Next i
```

Then fill the signal data to the 6th (F) and the next columns.

```
        For s = 0 To serCnt
            sht.Cells(1, s + 6).Value = serNames(s)
            For i = 0 To valCnt
                sht.Cells(i + 2, s + 6).Value = data(s, i)
            Next i
        Next s

    Else
        MsgBox (msg)
    End If
```

Let the sub-routine *RecordAndAnalyze* be repeated after the next 5 seconds with newer data and so on until the Excel file is closed. Add the code

```
    ScheduleTime = Now + TimeValue("00:00:05")
    Application.OnTime ScheduleTime, "RecordAndAnalyze"

End Sub
```

to the end of the sub-routine *RecordAndAnalyze* and create a sub-routine on the file-close event that will cancel the last schedule.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

Application.OnTime EarliestTime:=ScheduleTime, ...
                   Procedure:="RecordAndAnalyze", ...
                   Schedule:=False

End Sub
```

Appendix C contains the complete source code of the Microsoft Excel Visual Basic script.

Using the PC Master Software OnRecorderDone Event

In the previous example the sub-routine *RecordAndAnalyze* is called periodically each 5 seconds. Another way to handle the PC master software recorder data is to use the *OnRecorderDone* event. This event is called each time the currently-selected recorder finishes downloading of the new data. To assign a handler to this event in Visual Basic, declare the PC master software ActiveX object with the *WithEvents* keyword.

```
Public WithEvents pcm As McbPcm
```

and then create the event-handler routine:

```
Private Sub pcm_OnRecorderDone()

  /* any code */

End Sub
```

Extension

Methods similar to creating Control Pages can be used also for other HTML pages within the PC master software:

- Detail View pages
- Algorithm Block Description pages

Conclusion

The aim of this Application Note is to describe and to demonstrate how to create advanced PC master software Control Pages, which are intended to monitor and to control embedded applications. Therefore, the complete methodology with all required information is provided. The description is illustrated by real examples, with code samples.

A full real-life example is also provided. The example page is divided into two frames. The usage of PC master software functions from an HTML page is shown in one frame and the usage of PC master software functions from Microsoft Excel is shown in the second frame.

Appendix A

```
/*****************************************************************************
*
* Motorola Inc.
* (c) Copyright 2001 Motorola, Inc.
* ALL RIGHTS RESERVED.
*
*****************************************************************************
*
* File Name: lsm_demo.c
*
* Description: FIR adaptive processor running the signal prediction algorithm
*
* Modules Included:
*
*****************************************************************************/

#include "types.h"
#include "arch.h"
#include "periph.h"
#include "appconfig.h"
#include "gpio.h"
#include "itcn.h"
#include "sci.h"
#include "pcmaster.h"

#define MAX_ORDER       128

/* FIR adaptive processor */
Frac16  w[MAX_ORDER];  /* FIR filter coeficients  */
Frac16  x[MAX_ORDER];  /* input signal buffer     */
Frac16  y;             /* filter response         */
Frac16  d[MAX_ORDER];  /* desired response buff.  */
Frac16  e;             /* response  error         */
Frac16  mi;            /* step-size of adaptation */
UWord16 N;             /* filter order            */

/* Prediction params */
UWord16 D;             /* prediction delay        */

/* Signal generation */
Frac16  A1;            /* sine 1 amplitude           */
Frac16  A2;            /* sine 2 amplitude           */
Frac16  omg1;          /* sine 1 actual phase <-1,1) */
Frac16  omg2;          /* sine 2 actual phase <-1,1) */
Frac16  d_omg1;        /* sine 1 phase inc <-1,1)    */
Frac16  d_omg2;        /* sine 2 phase inc <-1,1)    */


void main (void)
{
   UWord32 j;
   UWord16 i;
```

```
/* PC master software initialization */
pcmasterInit();

/* ITCN initialization */
ioctl(ITCN,  ITCN_INIT_GPRS, NULL);   /* set GPR and IPR according to the */
ioctl(ITCN,  ITCN_INIT_IPR, NULL);    /* definition in appconfig.h */

/* SCI initialization */
ioctl(SCI_0, SCI_INIT, NULL);

archEnableInt();                       /* enable all maskable interrupts */

 /* initialization */
N = 16;
mi = __fixed2int(0.001);
D = 10;
A1 = __fixed2int(0.5);
A2 = __fixed2int(0.4);
d_omg1 = __fixed2int(0.06);
d_omg2 = __fixed2int(0.09);

while(1)
{

   /* recorder routine call -> simulate periodical call
   of pcmasterRecorder routine e.g. in timer interrupt */

   pcmasterRecorder();

   /* delay */
   for (j=0 ; j<10000 ; j++ );


     /* signal generation */
     /* ----------------- */

     /* shift of input buffers x and d */
     for (i=N-1 ; i > 0 ; i--)
     {
       x[i] = x[i-1];
     }
     for (i=MAX_ORDER-1 ; i > 0 ; i--)
     {
       d[i] = d[i-1];
     }

     /* new input sample */
     x[0] = d[D];

     /* new desired response sample */
     omg1 = omg1 + d_omg1;
     omg2 = omg2 + d_omg2;
     d[0] = __mult(A1,fcnSinPIx(omg1)) + __mult(A2,fcnSinPIx(omg2));
```

```
        /* LMS adaptive algorithm */
        /* ---------------------- */

        /* FIR filter processing */
        y = 0;
        for (i=0 ; i<N ; i++ )
        {
          y += __mult(w[i],x[i]);
        }

        /* result error */
        e = d[0] - y;

        /* filter coefficients update */
        for (i=0 ; i<N ; i++ )
        {
          w[i] = w[i] + 2*__mult(__mult(mi,e),x[i]);
        }

   }

}
```

Appendix B

```
<!*************************************************************************>
<!                                                                         >
<! Motorola Inc.                                                          >
<! (c) Copyright 2001 Motorola, Inc.                                      >
<! ALL RIGHTS RESERVED.                                                   >
<!                                                                         >
<!*************************************************************************>
<!                                                                         >
<! File Name: control_panel.xls                                           >
<!                                                                         >
<! Description: control tables to adjust application parameters           >
<!                                                                         >
<! Modules Included:                                                      >
<!                                                                         >
<!*************************************************************************>

<html>

  <head>
    <meta http-equiv="content-type" content="text/html;charset=iso-8859-1">
    <meta name="generator" content="Adobe GoLive 4">
    <title>Control Page</title>
    <link rel="stylesheet" href="LMS.css" type="text/css">
  </head>

  <body>

  <script language="VBScript">

Sub ReadSettings()

  Dim succ,v,tv,retMsg

  succ = pcm.ReadVariable("A1",v,tv,retMsg)

  If succ then
    A1.Value = tv
  else
    A1.Value = retMsg
  End If

  succ = pcm.ReadVariable("d_omg1",v,tv,retMsg)

  If succ then
    d_omg1.Value = tv
  else
    d_omg1.Value = retMsg
  End If

  succ = pcm.ReadVariable("A2",v,tv,retMsg)

  If succ then
    A2.Value = tv
  else
    A2.Value = retMsg
  End If
```

```
  succ = pcm.ReadVariable("d_omg2",v,tv,retMsg)

  If succ then
    d_omg2.Value = tv
  else
    d_omg2.Value = retMsg
  End If

  succ = pcm.ReadVariable("D",v,tv,retMsg)

  If succ then
    D.Value = tv
  else
    D.Value = retMsg
  End If

  succ = pcm.ReadVariable("N",v,tv,retMsg)

  If succ then
    N.Value = tv
  else
    N.Value = retMsg
  End If

  succ = pcm.ReadVariable("mi",v,tv,retMsg)

  If succ then
    mi.Value = tv
  else
    mi.Value = retMsg
  End If

End Sub


Sub WriteValue

  Dim succ,retMsg

  succ = pcm.WriteVariable(Window.Event.SrcElement.Name,Window.Event.SrcElement.Value,...
                           retMsg)

  If succ then
    txtError.InnerText = ""
  else
    txtError.InnerText = retMsg
  End If

End Sub


Sub WriteValueOnEnter

  If chr(Window.Event.KeyCode) = VBCR Then
    WriteValue
  End If

End Sub
```

```
</script>
  <object classid=clsid:48A185F1-FFDB-11D3-80E3-00C04F176153 height=1
          id=MCB_PC_Master_Services1 name=pcm width=1></object>
  <table border="0" cellpadding="5" cellspacing="0">
    <tr>
      <td valign="top">
        <table bgcolor="#00aacc" border="8" bordercolordark="#006688"
               bordercolorlight="#00ddff" cellpadding="4" cellspacing="0" width="100%">
          <tr>
            <td align="center" valign="middle">
               <font size="4"><b>Signal Generation</b></font>
            </td>
          </tr>
          <tr>
            <td>
              <table border="0" cellpadding="2" cellspacing="2" width="100%">
                <tr>
                  <td colspan="2" align="center"><b>Harmonic I</b></td>
                </tr>
                <tr>
                  <td>amplitude:</td>
                  <td><input type="text" name="A1" size="7" onchange="WriteValue"
                       onkeydown="WriteValueOnEnter"></td>
                </tr>
                <tr>
                  <td nowrap>phase inc.:</td>
                  <td nowrap><input type="text" name="d_omg1" size="7"
                             onchange="WriteValue" onkeydown="WriteValueOnEnter"> rad/pi
                  </td>
                </tr>
              </table>
            </td>
          </tr>
          <tr>
            <td>
              <table border="0" cellpadding="2" cellspacing="2" width="100%">
                <tr>
                  <td colspan="2" align="center"><b>Harmonic II</b></td>
                </tr>
                <tr>
                  <td>amplitude:</td>
                  <td><input type="text" name="A2" size="7" onchange="WriteValue"
                       onkeydown="WriteValueOnEnter"></td>
                </tr>
                <tr>
                  <td nowrap>phase inc.:</td>
                  <td nowrap>
                     <input type="text" name="d_omg2" size="7" onchange="WriteValue"
                      onkeydown="WriteValueOnEnter"> rad/pi
                  </td>
                </tr>
              </table>
            </td>
          </tr>
        </table>
      </td>
    </tr>
    <tr>
      <td>
        <table bgcolor="#00aacc" border="8" bordercolordark="#006688"
               bordercolorlight="#00ddff" cellpadding="4" cellspacing="0" width="100%">
```

```
          <tr>
            <td align="center" valign="middle">
               <font size="4"><b>Processing</b></font></td>
          </tr>
          <tr>
            <td>
              <table border="0" cellpadding="2" cellspacing="2" width="100%">
                <tr>
                  <td>prediction delay:</td>
                  <td nowrap><input type="text" name="D" size="7" onchange="WriteValue"
                              onkeydown="WriteValueOnEnter"> samples</td>
                </tr>
              </table>
            </td>
          </tr>
        </table>
      </td>
    </tr>
    <tr>
      <td>
        <table bgcolor="#00aacc" border="8" bordercolordark="#006688"
               bordercolorlight="#00ddff" cellpadding="4" cellspacing="0" width="100%">
          <tr>
            <td align="center" valign="middle" nowrap>
               <font size="4"><b>LMS Adaptive Algorithm</b></font></td>
          </tr>
          <tr>
            <td>
              <table border="0" cellpadding="2" cellspacing="2" width="100%">
                <tr>
                  <td nowrap>FIR processor order:</td>
                  <td><input type="text" name="N" size="7" onchange="WriteValue"
                       onkeydown="WriteValueOnEnter"></td>
                </tr>
                <tr>
                  <td nowrap>step-size:</td>
                  <td nowrap><input type="text" name="mi" size="7" onchange="WriteValue"
                              onkeydown="WriteValueOnEnter"></td>
                </tr>
              </table>
            </td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
  <tr>
  <center>
    <font color="red" id=txtError></font>
  </center>
  <script language="VBScript">
    ReadSettings
  </script>

</body></html>
```

Appendix C

```
í*************************************************************************

í Motorola Inc.
í (c) Copyright 2001 Motorola, Inc.
í ALL RIGHTS RESERVED.

í*************************************************************************

í File Name: lms.xls

í Description: capture of application data to third worksheet

í Modules Included: ThisWorkbook, Module1

í*************************************************************************

'******************
'*  ThisWorkbook:  *
'******************

Private Sub Workbook_Open()

    Set pcm = CreateObject("MCB.PCM")

    ScheduleTime = Now + TimeValue("00:00:05")
    Application.OnTime ScheduleTime, "RecordAndAnalyze"

End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)

    Application.OnTime EarliestTime:=ScheduleTime, Procedure:="RecordAndAnalyze", ...
                       Schedule:=False

End Sub

'******************
'*    Module1:    *
'******************

Public pcm As McbPcm
Public ScheduleTime As Variant

Sub RecordAndAnalyze()

    Dim succ As Boolean
    Dim sht As SheetData

    Set sht = Worksheets("Data")

    ' Spectral Function
    succ = pcm.ReadVariable("N", N, tv, msg)

    If succ Then

        succ = pcm.ReadMemory("w", 2 * N, data, msg)

        If succ Then

            ' clear data
            sht.Range("A2:A129").ClearContents
```

```
            ' returned array bounds
            serCnt = UBound(data)

            ' fill data
            For s = 0 To (serCnt \ 2)
                MSB = data(2 * s + 1)
                LSB = data(2 * s)
                If MSB >= 128 Then
                    sht.Cells(s + 2, 1).Value = (256 * MSB + LSB - 65535) / 32768
                Else
                    sht.Cells(s + 2, 1).Value = (256 * MSB + LSB) / 32768
                End If
            Next s

        Else ' something failed
            MsgBox (msg)
        End If

    Else ' something failed
        MsgBox (msg)
    End If


    ' Signals & Statistics
    succ = pcm.GetCurrentRecorderData_t(data, serNames, tbase, msg)

    If succ Then

        ' clear columns
        sht.Range("E:H").ClearContents

        ' returned array bounds
        serCnt = UBound(data, 1)
        valCnt = UBound(data, 2)

        ' timebase valid ?
        If tbase > 0 Then
            sht.Cells(1, 5).Value = "time [sec]"
        Else
            sht.Cells(1, 5).Value = "index"
            tbase = 1
        End If

        ' fill X-axis column
        For i = 0 To valCnt
            sht.Cells(i + 2, 5).Value = i * tbase
        Next i

        ' fill data
        For s = 0 To serCnt
            sht.Cells(1, s + 6).Value = serNames(s)
            For i = 0 To valCnt
                sht.Cells(i + 2, s + 6).Value = data(s, i)
            Next i
        Next s

    End If

    ScheduleTime = Now + TimeValue("00:00:05")
    Application.OnTime ScheduleTime, "RecordAndAnalyze"

End Sub
```

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

MOTOROLA

AN2263/D
Rev. 0
12/2002