Overview

Introduction

ModCon is short for MODular CONtroller. At the heart of the ModCon board is a Freescale 16-bit fixed-point microcontroller – the MC9S12A512. It has 512 KB of non-volatile storage (Flash memory), 14 KB of RAM, and numerous peripherals.

The ModCon board

The ModCon is used in the subjects:

- 48540 Signals and Systems
- 48434 Embedded Systems
- 48560 Analog and Digital Control.
The basic idea behind using the ModCon is shown in the diagram below:

Windows is a non-real-time operating system, and even though PCs are fast, Windows does not guarantee execution of code within a prescribed time. Consequently, the real-time task of controlling a system is given to the ModCon board. The ModCon acts as a stand-alone controller that is able to carry out tasks in real-time. The ModCon “talks” to a PC via a USB connection so that Windows can provide a graphical user interface (GUI) for us.

Controller Design Methodology

To perform control on a real system, there are two steps:

1. Characterize the system, i.e. devise a transfer function for the linear region of the system, and investigate non-linear behaviour and limits.

2. Control the system using either an analog or digital controller.

ModCon Modes

The ModCon board has two modes to facilitate the design methodology. They are:

1. Function Generator – the ModCon acts as a function generator so that various tests on the system-to-be-controlled can be carried out, such as step response and frequency response.

2. Controller – the ModCon performs a user algorithm that can take up to 4 inputs and produce up to 2 outputs, in real-time.

ModCon PC Interface

The ModCon PC interface provides a GUI to control the ModCon board.
Freescale CodeWarrior

Freescale CodeWarrior is an integrated development environment that is used to create programs to run on the ModCon board. In Analog and Digital Control, you use the compiler to generate a control algorithm which can then be downloaded to the ModCon board using the ModCon PC interface.
ModCon Connections

External Connections

The ModCon sits in a PC disk drive cradle that provides 4mm sockets that connect to the ModCon’s analog channels.

There are 8 input channels and 4 output channels.

The input channels are differential channels, so both positive (red) and negative (black) connections must be made.

The output channels are single-ended, so only one common connection is required on the output (all green terminals are at the same potential).

An example connection is shown below, where two input channels are used (red and yellow), one output channel (blue), and they are all referenced to a single common (green).

Internal Connections

The ModCon is attached to the PC via an internal USB cable.
ModCon PC Interface

Overview

The ModCon PC Interface will start up and display an interface appropriate to the current mode of the ModCon.

The ModCon PC Interface has the following components:

ModCon Number

The ModCon number simply tells you the serial number of the ModCon board you are using.

ModCon Info

The ModCon information area tells you the version number of the ModCon firmware, the mode of the ModCon board (Function Generator or Controller), the Controller mode (Basic, Intermediate, Advanced, Automatic), the Control Period (in milliseconds) and the microcontroller’s utilization.
Data Acquisition

The Data acquisition area allows you to start and stop the data acquisition process. When stopped, you can save the viewed data to a comma separated variable (CSV) file that can be imported into many other programs, such as Excel.

Graphs

The Graphs area gives a real-time display of various analog quantities. There are 4 inputs and 2 outputs. The graphs interface is based around a Digital Storage Oscilloscope (DSO) so it should be a familiar interface.

Time Used

The time is the time that the ModCon board has been operating, in hh:mm:ss format.

ModCon Function

This area changes to reflect the current mode of operation of the ModCon. In Function Generator mode, it provides an interface to set up parameters such as wave type, amplitude, offset etc. In Controller mode, and depending on the control mode, it provides you with the ability to enter control parameters (and algorithms) and to turn the controller on and off.
ModCon Number

The ModCon number simply tells you the serial number of the ModCon board you are using. This could be handy if there is a problem with the hardware, as it enables staff to keep track of any failures or modifications made to the individual boards.
ModCon Info

The ModCon Info area tells you:

- the version number of the ModCon firmware
- the mode of the ModCon board (Controller or Function Generator)
- the Controller mode (Bas, Intermediate, Advanced, Automatic)
- the Control Period, in milliseconds.
- utilization statistics.

Version

The version number is for reference purposes only. The firmware inside the ModCon board is automatically updated via the ModCon PC interface every time it starts.

ModCon Mode

The ModCon Mode gives you the choice of either Controller or Function Generator. The Mode is saved in non-volatile memory so that the ModCon always starts up in the last-used mode.
The Control Mode has four options: Basic, Intermediate, Advanced and Automatic.

**Basic** control is for single-input single-output systems that only require first-order compensators. The coefficients in this mode are integers with respect to a base of 256. This mode is used in Signals and Systems.

**Intermediate** control is not yet implemented.

**Advanced** control allows you to enter a custom control algorithm written in C. It is the mode used in Analog and Digital Control.

**Automatic** is not yet implemented and is intended for demonstration purposes.

The Control Period is the time in milliseconds (ms) between control efforts. You can set the control period between the values shown to suit the system being controlled. The shorter the control period, the higher the microcontroller’s utilization (CPU time). For complicated control algorithms, this setting needs to be taken into account.
Utilization Statistics

The utilization statistics tell you how hard the microcontroller is working. For small control periods and complex algorithms, the utilization is high.

**It is important to keep the microcontroller’s utilization as low as practical – time is a finite resource!**

Update Button

The Update button sends the new control period to the microcontroller. This is the only way you can set the control period – you cannot set it in your C code. The control period is stored in non-volatile memory so that the ModCon always starts up with the last-used control period.
Data Acquisition

The Data Acquisition area lets you start and stop the acquisition of the ModCon’s input and output variables by the PC.

The Save button is only available once the acquisition has been halted. What you see on the graph is then saved to a comma separated variable (CSV) file.

The format of the CSV file is:

**Column 1**: Time

Starts at 0 and increments by the graph sample period in seconds (s).

**Columns 2-5**: Inputs 1-4

Actual input values in volts (V).

**Column 6-7**: Outputs 1-2

Actual output values in volts (V).

If you have renamed the input and output channels, then those names will be used as the column headings.
The Graphs area is a key area of the PC interface. It allows you to see the input and output variables of the ModCon displayed on a graph in real-time. The interface has been set up to mimic a digital storage oscilloscope (DSO) so that controls of the display will be familiar.

Options

Options to change the nature of the display are accessible via a pull-down list at the top-left or by buttons placed along the bottom of the Graphs area.
The vertical options apply to the currently selected channel (via the pull-down list) and allow you to change:

- whether the selected channel is displayed or not (i.e. on or off)
- the scale of the channel, in volts per division (V/div)
- the position of the channel, in volts (V)
- the name of the channel (up to 255 characters)
- the color of the channel (24-bit color)

The values option shows you the current value of each channel. It is handy for reading off steady-state values or very slowly changing values.

The names of the channels are displayed above the relevant value (the names of the channels are updated in the Vertical options).

The color of the value corresponds to the color of the channel.
M.14

Timebase

The timebase option allows you to change the horizontal time (seconds) per division. If the time per division is 0.5 s/div or larger, then the trigger mode changes to AUTO-ROLL. This value of entering AUTO-ROLL cannot be changed.

The timebase option also gives you information about the number of actual samples displayed, the sample rate, the sample resolution and the total time of the sample window.

Trigger

The trigger option allows you to change the trigger mode. The ModCon PC interface program will always start in Auto trigger mode. This is generally the best setting to begin with.

The modes for the trigger are:

**Stop** – do not acquire any samples.

**Auto** – Automatically trigger off the currently selected channel with the currently selected slope. The trigger point is the first division and this cannot be changed.

**Normal** – The trigger level (vertical value) can be set manually via the Level (V) control. The graph will update every time the selected channel crosses the trigger level with the selected slope.

**Single** – The graph will behave like the Normal trigger, but will only occur once. After triggering the trigger mode reverts to Stop. This is useful for
acquiring on-off waveforms.

**Roll** – The graph will roll from right to left. Useful for slowly varying waveforms where it is inconvenient to wait for a trigger event and subsequent acquisition. The display will AUTO-ROLL if the timebase is set to 0.5 s/div or larger.

**Update** – The graph will cyclically update from left to right, overwriting old values with new values.

**Graph**

The graph Type allows you to change the display between single, dual and quad modes.

The grid can be turned on and off using the Grid On checkbox.

You can arbitrarily assign the channels to a position on the dual and quad graph types by first selecting the Channel, then choosing where it goes using the Dual and Quad drop-down lists.
ModCon Modes

Overview

There are two modes of operation for the ModCon. One is a stand-alone function generator (output only), the other is a controller.

Function Generator

This mode provides function generator outputs, as well as measuring all inputs. It is used for system characterization.

ModCon Controller

This mode provides for custom control of the plant. You need to write a control algorithm in C and download it to the ModCon board for this mode to work.
The Function Generator has been incorporated into the ModCon to facilitate system characterization.

The ModCon provides a dual, independent output, function generator which is capable of generating a typical variety of waveforms.

These waveforms can be used to perform step responses, frequency responses and other functions.

**Output Channel**

Use the output channel pull-down list to choose the currently active output channel of the function generator. The parameters of the selected channel will be updated in the Entry and Settings group boxes.
Entry

The Entry group box reflects the current settings that will be applied to the currently selected output channel when you push the Update button. The parameters you can set are:

**Function** – choose between Sine, Square, Triangle, Sawtooth, Noise and Arbitrary (Arbitrary is not yet implemented).

**Frequency** - choose between 0.1 Hz and 100 Hz in 0.1 Hz steps.

**Amplitude** – choose between 0 V and 10 V in 0.1 V steps.

**Offset** – choose between -10 V and 10 V in 0.1 V steps.

Settings

The Settings group box reflects the actual settings that have been applied to the currently selected output channel. The Settings also tell you whether the function generator output is on or off. The Start / Stop button in the bottom right-hand corner allows you to turn the function generator on or off.
ModCon Controller

The ModCon Controller has four modes of operation, which are chosen using the pull-down list in the ModCon Info box.

Control Mode

The Control Mode has four options: Basic, Intermediate, Advanced and Automatic.

**Basic** control is for single-input single-output systems that only require first-order compensators. The coefficients in this mode are integers with respect to a base of 256. This mode is used in Signals and Systems.

**Intermediate** control is not yet implemented.

**Advanced** control allows you to enter a custom control algorithm written in C. It is the mode used in Analog and Digital Control.

**Automatic** is not yet implemented and is intended for demonstration purposes.
The ModCon Basic Controller is for use in single-input, single-output (SISO) systems. It has very limited functionality – it can only implement two first-order compensators. One compensator is in the feedforward path, the other in the feedback path. The control algorithm is fixed by the topology shown, and only uses Input 1 as the input, and Output 1 as the output. The controller coefficients are fixed-point and assume a base of 256 (i.e. all numbers entered will be internally divided by 256 by the ModCon when it performs its control algorithm).
The ModCon Advanced Controller is for use in multiple-input, multiple-output (MIMO) systems. It is extremely flexible – the control algorithm is written in C and uploaded to the ModCon board. Control algorithms can be linear difference equations, non-linear filters, fuzzy controllers, neural networks, or anything that can be implemented on the microcontroller within the control period.

Reference

The R button is used to bring up the Reference control panel. The reference can be adjusted to between -10 V and 10 V in 0.01 V steps.
Algorithm

The Algorithm button is used to specify the location of an S19 file for uploading into the ModCon board.

An S19 file is generated by CodeWarrior and contains the microcontroller’s object code (in ASCII format).

Press the Choose… button to locate the S19 file that you wish to upload to the ModCon.

The S19 file is located in the bin folder where your CodeWarrior project files are. It normally has the name control.abs.s19.
If you press *Ok*, the microcontroller’s firmware will be updated with your control algorithm. It is stored in non-volatile memory (Flash) in the ModCon so that the next time you start work the algorithm is already loaded. The chosen file name is also stored on the PC, so you don’t have to keep choosing the file name if you are using and modifying the same algorithm.

**Be careful – since the ModCon board is shared, other users may have uploaded a different algorithm to yours. Always check initially that you are running the correct algorithm.**

If you press *Cancel* the selected algorithm will not be updated.

### Parameters

The *Parameters* button brings up a panel that allows you to change the control parameters.

There are 8 floating-point control parameters, labelled CP₁ to CP₈.

The meaning of the control parameters is determined entirely by the control algorithm that has been uploaded to the ModCon.

The parameters are generally used to change the coefficients and gains in the control algorithm, without having to reprogram new firmware.
Starting and Stopping

The Start / Stop button allows you to toggle the state of the controller on and off. The controller state (on or off) next to the button is updated accordingly.
Programming a Control Algorithm

The microcontroller used in the ModCon board has a different instruction set to a PC (as most other chips do). Therefore, we need to compile code using a cross-compiler – one that runs on a PC but compiles for a different platform. The cross-compiler we use is Freescale’s CodeWarrior. The resulting object code then needs to be uploaded into the ModCon.

The control algorithm for the advanced controller is up to you.

Basically, once set up, there are four steps to making an algorithm:

1. Open your controller project file with CodeWarrior.
2. Open your C control file in CodeWarrior’s code editor.
3. Write your control algorithm in C.
4. Make the controller project (compile your code).
5. Download the new control algorithm to the ModCon board.

Detailed steps are given below.

Initially, you need to do the following:

Download the CodeWarrior project files from the web and unzip them to a convenient folder. The zip file has folder information so you should see a folder structure as shown.

The control algorithm is programmed in the C language and compiled using Freescale’s CodeWarrior.
Thereafter, perform the following steps:

1. Double-click the file called Control.mcp. This is the CodeWarrior project file. CodeWarrior should start.

2. When CodeWarrior starts, you see a mostly blank screen with a project pane on the left. Inside the project pane there are three files visible:

   readme.txt – you should obviously read this.

   main.c – the main file of the project.

   control.c – the place where you implement your control algorithm.

   Double-click control.c – the file is loaded and displayed to the right in a code editor. You may maximize this child window to increase the editor size.
3. Write C code for the Control_DoControl function that implements your control algorithm. For help on programming the control algorithm, see the help file called Programming Details.

4. Make the CodeWarrior project by pressing the shortcut key F7, or by choosing Project | Make from the pull-down menu. If there are no errors, CodeWarrior will generate an S19 file with your algorithm in it, ready for uploading to the ModCon board.
5. On the ModCon Advanced Controller, click on the Algorithm button and choose the recently compiled S19 file to download. Click OK and your control algorithm is downloaded to the ModCon ready for testing!
Programming Details

The ModCon firmware hides most of the details of actually performing the control for you. At a high level, the ModCon sets up a periodic timer at intervals corresponding to the control period. Then, periodically, it:

1. Samples the analog channels with an analog to digital converter, and scales the inputs to obtain units of volts.

2. Calls your control function called Control_DoControl().

3. Scales and hard-limits any output values from your control algorithm, and updates the output channels via a digital to analog converter.

4. Updates a history of all control variables.

5. Sends the values of all inputs and outputs to the PC for display purposes.

Most of this complexity is hidden in a library file in your CodeWarrior project. You only have access to the control.c file which has one function called Control_DoControl(). As shown above, it is called periodically by the main program.

Variables

The control.c file defines several variables that you should become familiar with:

```c
// Reference
TFloat Reference;

// Input variables
double X[NB_INPUT_CHANNELS][NB_PAST_VALUES];

// Output variables
double Y[NB_OUTPUT_CHANNELS][NB_PAST_VALUES];

// Intermediate variables
double Z[NB_INTERMEDIATE_VARIABLES][NB_PAST_VALUES];

// Advanced control parameters
TFloat CP[NB_CONTROL_PARAMETERS];
```

ModCon 2008
Short-cuts to these variables are also defined, which makes your code easier to read, and hides structure access details for the custom struct called TFloat (the “.d” for the Reference and CP variables)

// Short-cut macros for variables
#define Ref Reference.d
#define CP1 CP[0].d
#define CP2 CP[1].d
#define CP3 CP[2].d
#define CP4 CP[3].d
#define CP5 CP[4].d
#define CP6 CP[5].d
#define CP7 CP[6].d
#define CP8 CP[7].d
#define X1 X[Ch1]
#define X2 X[Ch2]
#define X3 X[Ch3]
#define X4 X[Ch4]
#define Y1 Y[Ch1]
#define Y2 Y[Ch2]
#define Z1 Z[0]
#define Z2 Z[1]
#define Z3 Z[2]
#define Z4 Z[3]

These are described below:

Reference Variable

Ref – This is the controller’s reference signal. The reference value can be updated using the PC interface.

Input, Output and Intermediate Variables

X – these are the input signals to the ModCon. NB_INPUT_CHANNELS equals 4.

Y – these are the output signals from the ModCon. NB_OUTPUT_CHANNELS equals 2.

Z – these are intermediate signals available for free use by you. NB_INTERMEDIATE_VARIABLES equals 4.
The variables $X$, $Y$, and $Z$ are special. Notice that they are two-dimensional arrays. The first array dimension specifies the channel number. The second array dimension specifies the past value index. The array is organised so that the second subscript corresponds to the delay. Short-cuts to the elements of the array have been defined to make your C code easier to read. For example, if you wish to use an input value from Input 1 which occurred 3 control periods ago, you would use it as $X1[3]$. Similarly, the current output value on Output 1 would be referenced as $Y1[0]$. Similarly for the other variables.

For example, to translate the difference equation:

$$y[n] = 0.1 \times x[n] + 0.9 \times y[n-1]$$

you would do:

$$Y1[0] = 0.1 \times X1[0] + 0.9 \times Y1[1];$$

(assuming that $X$ was on Input 1, and $Y$ was on Output 1).

The first array dimension on the $Z$ variable does not correspond to a physical analog channel.

**Control Parameter Variables**

$CP$ – these are the control parameters. The control parameter values can be updated using the PC interface. $NB\_CONTROL\_PARAMETERS$ equals 8.

**Other Variables**

You are free to define any other variables you require. The advantage of using $X$, $Y$, and $Z$ is that the ModCon automatically updates past values for you. If you define new variables that need a history, you will have to update them yourself in the control algorithm.
Algorithm

An example control algorithm is shown below.

```c
// ----------------------------------------
// Control_DoControl
// Performs a control algorithm from input to output
// Input: none
// Output: none
// Conditions: Assumes that the ADC has been set up
void Control_DoControl(void)
{
    // Ball and beam
    X1 X2
    Ball
    // Ref + / \
    Z1 +-----+ Z2 + / \ Z3 +-----+ Y +----- + Angle +----- +
    // Position
    // ------| ------| C1 |------| ------| C2 |------| Beam |------| Ball |------|
    // / +-----+ / +-----+ / +-----+ +----- + +----- +
    //              | Beam Angle (Ch1)
    //              +-----------------------------+
    //              |
    //              Z4 +----- Ball Position (Ch2)
    //  +----------------------------------------------------------+
    //  |
    //  |
    //  CP0 = 1.625        CP4 = -13.6
    //  CP1 = -1.08712     CP5 = 13.5184
    //  CP2 = 0.513        CP6 = 0.932
    //  CP3 = 0            CP7 = 0
    // Ball and Beam control
    // Lowpass filter the ball position to remove noise
    Z4[0] = 0.1 * X2[0] + 0.9 * Z4[1];
    // Ball position error = Reference - Actual Position
    Z1[0] = Ref – Z4[0];
    // Calculate control effort for ball dynamics
    Z2[0] = CP5 * Z1[0] + CP6 * Z1[1] + CP7 * Z2[1];
    // Beam angle error = Desired Beam Angle - Actual Beam Angle
    Z3[0] = Z2[0] - X1[0];
    // Calculate control effort for motor and beam dynamics
    Y1[0] = CP1 * Z3[0] + CP2 * Z3[1] + CP3 * Y1[1];
}
```

Time

Time is the one quantity we all wish we had more of, and the microcontroller is no exception. Be aware that the floating-point operations in the control algorithm are implemented using a software library, since the microcontroller in the ModCon board has a fixed-point CPU. Emulating floating-point operations is expensive in terms of CPU time. Try to limit the number of operations you perform to get a reasonable utilization of the CPU. Erratic behaviour can result if you run out of time!

*Try and keep your code short and fast so that you don’t run out of time*

*Windows and Excel are registered trademarks of Microsoft Corporation in the United States and other countries*