AN1775

Expanding Digital Input with an A/D Converter

By Brad Bierschenk
Freescale Applications Engineering
Austin, Texas

Introduction

This application note documents a method of extending digital input using the analog-to-digital converter (ADC) of a microcontroller unit (MCU).

Many MCU applications require digital input and arbitration. For example, determining which key of a keypad was pressed. This is commonly done by arranging switches in a matrix configuration, connecting to a series of digital input pins, and reading a digital input data register to determine which key was pressed. While this method is easily implemented, it does require the use of an MCU's parallel port pins.

Some applications require all available bidirectional or input-only pins for other purposes. In such a case, an alternate method of arbitrating keypresses is desired. By using the ADC of an MCU connected to a resistor ladder, user input can be more efficiently processed.

Freescale Semiconductor, Inc.

freescale™ semiconductor

© Freescale Semiconductor, Inc., 2004. All rights reserved.

© Motorola, Inc., 1998

For More Information On This Product,
Go to: www.freescale.com

Background

Dedicated Input

A microcontroller typically receives user input through digital input pins. The simplest implementation is a single switch directly connected to a digital input pin. This is easy to realize, but is not the most efficient use of resources, with one pin dedicated to one input. One port data bit represents the state of one switch.

Matrix Input

Another method uses a keypad, a common element in embedded systems. These are ordinarily arranged in a matrix, as shown in Figure 1. In this case, the byte value of an entire port data register can be polled to determine which key was pressed. This is more efficient, as a 4 x 4 keypad can interface 16 keys with eight input pins.

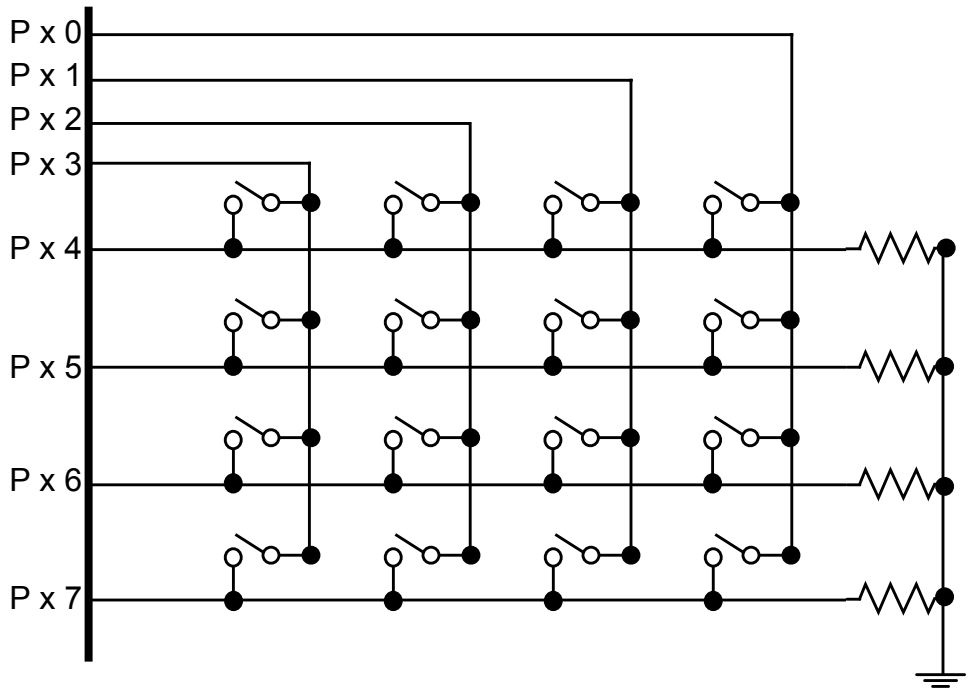

Figure 1. Resistor Matrix Keypad Using Parallel Port Pins

Freescale Semiconductor, Inc.

ADC Alternative

In many cases, input pins are at a premium. One can't always freely assign input pins to the function of user input. A more efficient use of microcontroller resources can be devised. One common feature of many Freescale MCUs is the analog-to-digital converter, or ADC.

The ADC of a Freesca;e MCU usually features four to eight channels of analog input, which is compared with a reference voltage and converted to an 8-bit digital value. When a resistor ladder is connected to an analog input through switches in each segment, the conversion result can be used to arbitrate an input. This allows many keys to be interfaced with one input pin, with only a little more software overhead. **Figure 2** shows such an implementation.

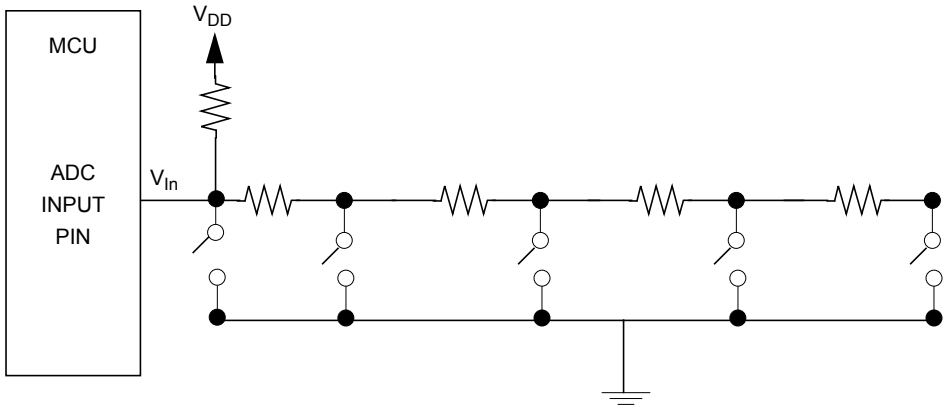

Figure 2. Resistor Ladder Keys Using an Analog Input Pin

Implementation

ADC Operation

An MCU ADC typically has 8-bit *precision*. This means there are 2^8, or 256, distinguishable A/D inputs, including 0. The analog inputs are converted to a binary number, which represents the magnitude of the input voltage in relation to a reference voltage.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

The *range* of an ADC is the difference between its high and low reference voltages. This means an analog input between V_{REFH} and V_{REFL} will convert to an 8-bit number, with V_{REFL} converting to $00 and V_{REFH} converting to $FF.

The *resolution*, defined as the range divided by the precision, defines the analog step that a change in one least significant bit (LSB) represents.

In the case of a 5-volt, 8-bit ADC, the resolution is 5/255 (volts), or 19.6 mV. This means that a change in one LSB in the ADC data register reflects a change of about 20 mV at the analog input.

Resistor Ladder Voltage Divider

Consider a resistor ladder connected to an ADC input, as shown in Figure 3. Because this arrangement is a voltage divider, each segment in the ladder can alter the voltage at the input when grounded. If switches are provided at each segment, one can selectively ground that segment, altering the composition of the divider, and thus altering the voltage presented to the ADC pin.

In this way, software can determine which switch in the ladder was selected by reading the resulting A/D data value. The resistor R0 acts as a pullup to maintain V_{DD} on the analog input line while no keys are active. Thus, a conversion value of $FF indicates that no key has been pressed.

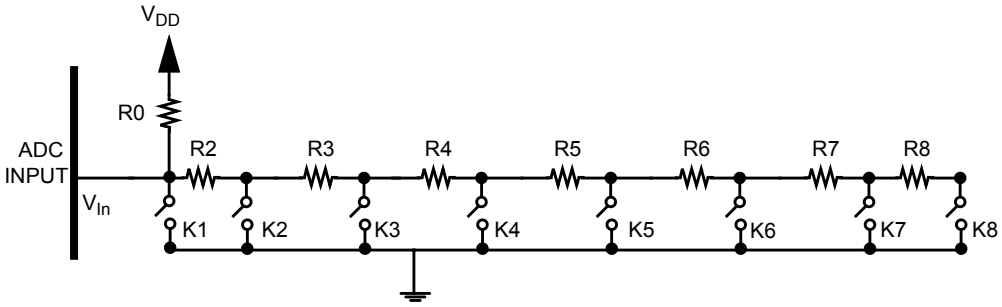

Figure 3. Digital Input Alternative

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

Considerations

Using this method, one can theoretically connect 255 input switches to one ADC pin. However, there are many potential sources of inaccuracies, which make it impractical to connect so many key inputs. One should account for some error padding.

Resistor Precision

One inaccuracy is provided by the resistors themselves. Resistors are categorized according to their variance from a labeled value. The application should be tolerant to the precision of the resistors being used (typically, 1 percent or 5 percent). Also, a calculated resistor value might not be a commonly available value, so the user should plan for a range of resistor values.

ADC Accuracy

Typically, an 8-bit ADC is accurate within two least significant bits. This should be accounted for as well. The best way to allow for these tolerances is to assign to each key switch a range of resulting A/D data register values.

If an analog input falls within a particular range, one can determine that the key was pressed. By adjusting the range of ADC results which represent a given keypress, the user can change the error margin for the application.

To ensure the best ADC accuracy, the full range of the converter should be used. In cases where the high reference is not variable, it is typically fixed at the operating voltage.

A spreadsheet is a good way to determine resistor values and A/D result ranges. An example of computing values for **Figure 3** are shown in **Table 1**.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 1. Spreadsheet Calculations

V_{DD} =	5	Converter Resolution =			0.01961	(Vdd/255)			
# Keys =	8								
Voltage Steps =	0.625	(V_{DD}/# Keys)		R0 =	10000	(Pullup)			
								(Hex)	(Hex)
Key pressed	V_{In}	Req	Rn			Vinmin	Vinmax	ADDRmin	ADDRmax
No Key	5	0	10000			4.6875	5	EF	FF
1	0	0	0			0	0.3125	0	F
2	0.625	1429	1429			0.3125	0.9375	F	2F
3	1.25	3333	1904			0.9375	1.5625	2F	4F
4	1.875	6000	2667			1.5625	2.1875	4F	6F
5	2.5	10000	4000			2.1875	2.8125	6F	8F
6	3.125	16667	6667			2.8125	3.4375	8F	AF
7	3.75	30000	13333			3.4375	4.0625	AF	CF
8	4.375	70000	40000			4.0625	4.6875	CF	EF

Some notes on the spreadsheet:

- V_{In} is determined by decrementing the high voltage reference by the voltage step for each segment in the ladder. (See **Figure 4**.)
- Rn is the resistor value of the current key segment needed to form desired equivalent resistance Req.
- Req represents the equivalent resistance of ladder, including the current resistor (Rn) and excluding the pullup resistor (R0).
- Vinmin and Vinmax are the minimum and maximum voltages that can be arbitrated as a particular key. In this case, V_{In} +/–Vstep/2 was used.
- ADDRmin and ADDRmax are the ADC data register value range used to represent a given keypress. This range can be narrowed or widened to affect precision. In this case, the ranges were maximized, so no conversion result is undefined.
- This particular spreadsheet did not use resistor precision, but this could be considered to further pad the error.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

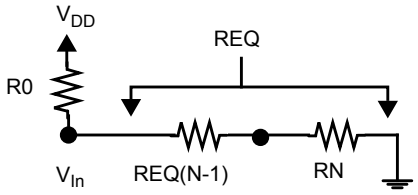

Figure 4. Voltage Divider

Constants

First, V_{DD} (or V_{REFH}, the high reference for the converter) and the number of keys are determined. The converter resolution is found by dividing V_{DD} by the converter precision. The number of voltage steps needed is V_{DD} divided by the number of key switches. An ADC should always be operated at its full range. The pullup resistor, R0, typically, should be chosen between 4.7 k (to limit current) to 22 k (to limit time constant).

Variables

Once these constant values are decided, a spreadsheet can be used to determine the resistor values needed in each segment of the divider ladder, according to the desired input voltages.

When using an ADC, always use the entire range of the converter ($V_{REFH} – V_{REFL}$). This is the reason for determining the voltage steps first (V_{DD} – n*step value) and then calculating the necessary resistor values to achieve these voltages through the divider.

For each segment in the ladder, the n^{th} segment's necessary resistor value needs to be calculated (Rn in the spreadsheet). When a key is pressed, the equivalent resistance of the included segment resistors forms a voltage divider with the pullup resistor.

Given the voltage desired at the ADC input (V_{In}), the user can determine the equivalent resistance needed to achieve that voltage by:

$$Req(n) = (V_{In}*R0)/(V_{REFH} – V_{In})$$

The resistor that will form the needed equivalent resistance with the other resistors in the ladder can be determined, as:

$$Rn = Req(n) – Req(n – 1)$$

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The only exception is key 1, which connects V_{SS} to the ADC input and needs no resistor.

By assigning a range of ADC conversion values to each key, the user can provide a fair amount of error padding. Considerations include resistor tolerance, ADC accuracy, and parasitic time constants. The range of conversion values for a particular keypress can be narrowed to improve the accuracy of the application. Or, for “quick and dirty” arbitration, keep the range as wide as possible.

Using a Single Resistor Value

The method presented here used different resistor values to produce equal voltage intervals. Another method would be to use the same resistance value for all segments in the divider. The disadvantage of using the same resistor values is that it greatly diminishes the effective range of conversions. Also, if resistances are kept equal, the voltage step between switches approaches the resolution of the ADC. Therefore, the error margin diminishes as more keys are added.

Programming Considerations

There are several ways to implement such an application. One thing to choose is whether to poll the ADC when desired or link a keypress to an interrupt source.

Some ADCs continuously convert once enabled, allowing a new value to be available every 32 clock cycles. Others do a single conversion when a register is written to and don’t do another conversion until the register is written to again.

A polling scheme can use a periodic timing source as a signal to poll. For example, the real-time interrupt (RTI) or timer overflow (TOF) interrupt can be used to scan the ADC input at a given rate.

Not linking a keypress to an interrupt source can cause timing problems and might miss a keypress. Careful consideration should be given to timing and voltage error requirements to determine if this method is appropriate.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

A Brief Example with the MC68HC705P6A MCU

The small code segment example that follows illustrates the software implementation of this method of keypress arbitration. The example was defined around the spreadsheet analysis example shown in **Table 1**.

The MC68HC705P6A (P6A) MCU features a 4-channel, 8-bit A/D converter. The P6A ADC uses continuous conversion, making a new value available every 32 internal clock cycles after being turned on.

The software example assumes a resistor ladder is connected to AD0 (A/D channel 0, port C, pin 6). The software is not intended to be a complete application.

This software starts the ADC, selects channel 0 for A/D conversions, then polls the ADC data register to determine if a key was pressed. The software uses a lookup table, with predefined maximum and minimum ADC values which represent a specific segment in the divider being grounded.

Once the key has been arbitrated, the RAM variable InKey will tell an application which key was switched most recently.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Code Listings

```
* ---------------------------------------------------------------------------
* EXPANDIO.ASM
* Written for the MC68HC705P6A microcontroller
* A code segment example to illustrate the use of the A/D
* converter for key input arbitration
*
* This simple example polls the AD0 channel, compares the
* conversion result to a lookup table, and determines
* which of 8 keys were pressed.
* ---------------------------------------------------------------------------

; Memory map equates
RAMSPACE        EQU     $50
ROMSPACE        EQU     $100

; A/D Registers
ADC             EQU     $1D                 ;A/D Data register
ADSC            EQU     $1E                 ;A/D Status and control

; ADSC Bits
CC              EQU     7                   ;Conversion complete flag
ADRC            EQU     6                   ;A/D RC oscillator enable
ADON            EQU     5                   ;A/D enable bit

* ---------------------------------------------------------------------------
; RAM Variables
* ---------------------------------------------------------------------------
                ORG     RAMSPACE            ;Start of user RAM
InKey           RMB     1                   ;Identifies the last key pressed
ADValue         RMB     1                   ;Stores the last ADC result

* ---------------------------------------------------------------------------
; Program code
; Simply loops, polling the A/D converter channel AD0
; and determining which key was pressed
* ---------------------------------------------------------------------------
                ORG     ROMSPACE            ;Start of user ROM
Start:
                LDA     #$20                ;Turn on A/D, select AD0 channel
                STA     ADSC

MainLoop:
                BRCLR   CC,ADSC,*           ;Wait for conversion complete
                LDA     ADC                 ;Get the result
                STA     ADValue             ;Record the result
                CLR     InKey               ;Clear the InKey variable
                CLRX                        ;Clear the offset
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
;Check the entries in the table, to find the ADC value range
;that corresponds to the ADC data register value.
KeyLoop:
              LDA     KeyTable+1,X          ;Check high range
              CMP     ADValue
              BLS     Match                 ;Within range
              LDA     KeyTable,X            ;Check low range
              CMP     ADVAlue
              BLS     Match                 ;Within range
              INCX                          ;Point to next table record
              INCX
              ;Increment the key value, when a match is made,
              ;the variable will contain the key that was pressed.
              INC     InKey
              BRA     KeyLoop

; At this point, InKey variable holds keypress information
; One can arbitrate the key press here. For this simple example
; we just repeat the main loop
Match:
              BRA     MainLoop


* ------------------------------------------------------------------------------
; Key lookup table. Holds the minimum and maximum ADC values
; which identify a particular key in the resistor ladder
* ------------------------------------------------------------------------------
KeyTable:
NoKey         FCB     $EF,$FF               ;No key pressed
Key1          FCB     $00,$0F
Key2          FCB     $0F,$2F
Key3          FCB     $2F,$4F
Key4          FCB     $4F,$6F
Key5          FCB     $6F,$8F
Key6          FCB     $8F,$AF
Key7          FCB     $AF,$CF
Key8          FCB     $CF,$EF


* ------------------------------------------------------------------------------
; Vector definitions
* ------------------------------------------------------------------------------
              ORG     $1FFE                 ;Reset vector
              FDB     Start
```

Freescale Semiconductor, Inc.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale Semiconductor, Inc.