Analog-to-Digital Conversion with the Neuron® Chip

January 1995 **LONWORKS® Engineering Bulletin**

This document describes some of the more popular analog to digital (A/D) conversion schemes available for use with the Neuron Chip. Included is the description and an example of the Neuron Chip's built-in Dualslope I/O object. The intent is not to provide an exhaustive survey of all conversion techniques but to provide an application-oriented reference for the user of the Neuron Chip.

For more specific information on A/D conversion techniques discussed in this application note, and on A/D conversion in general, refer to the list of references at the end of this document.

This document is divided into two major sections. The first section deals with traditional A/D conversion techniques and the specific attributes of each. Other aspects of analog data acquisition such as sample-and-hold circuits, analog multiplexers, and voltage references are also discussed in order to provide a foundation for what follows.

The second section of this engineering bulletin addresses the previously discussed A/D schemes in light of the capabilities and features of the Neuron Chip. Network performance issues relating to A/D conversion in a distributed control application are also covered, in order to make the developer aware of some of the common pitfalls associated with such systems.

The A/D design examples discussed include some off-the-shelf ICs in addition to circuits using more simplified external logic along with the Neuron Chip's I/O function blocks. As an aid to the designer, a sampling of different off-the-shelf A/D conversion ICs along with their features and attributes is included.

The Neuron Chip provides several I/O models which greatly simplify the task of interfacing to off-the-shelf A/D chips. These I/O model include the neurowire, bitshift, and I^2C I/O models. Refer to the *Neuron Chip Data Book* for more information on these I/O models

Background

Figure 1 shows the typical functional blocks found in most analog data acquisition designs. Different functional blocks may have varying complexities depending on the application. For example, processing low-level analog signals at high speed (conversion rate), might require more robust signal conditioning and high-speed conversion circuitry.

The majority of this engineering bulletin deals with the internals of the A/D block shown in figure 1. Some attention is given to the other functional blocks only to provide the reader with a working knowledge by which to support the A/D block design.

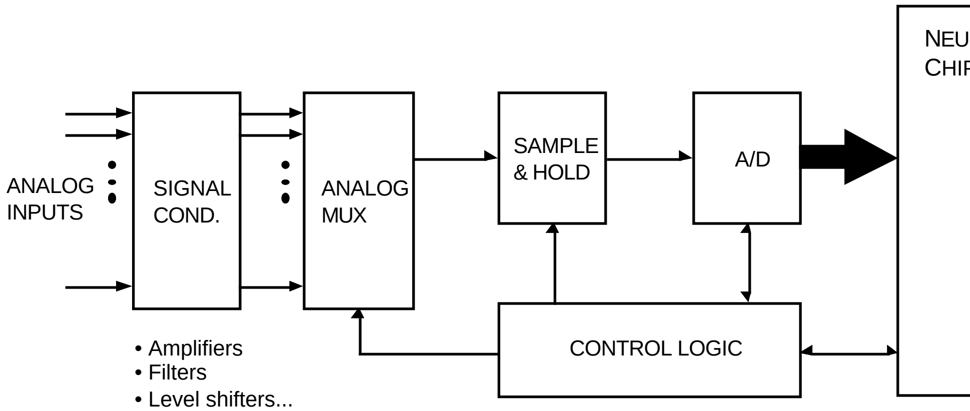

Figure 1. A typical analog data acquisition sub-system usually consists of several of the functional blocks shown.

Analog-to-digital conversion techniques can be divided into two distinct classes: *direct* and *indirect*.

In the *direct* conversion technique, the analog signal is continuously compared to the output of a D/A converter. The D/A converter's input is changed based on the results of the comparison, until the D/A converter's output matches the analog input signal. The input to the D/A converter is then the desired digital output.

Figure 2 illustrates the basic building blocks of a direct converter. The various direct conversion schemes differ in the way the input of the D/A converter is changed, namely the control process that uses the comparator output to generate the next digital input for the D/A converter.

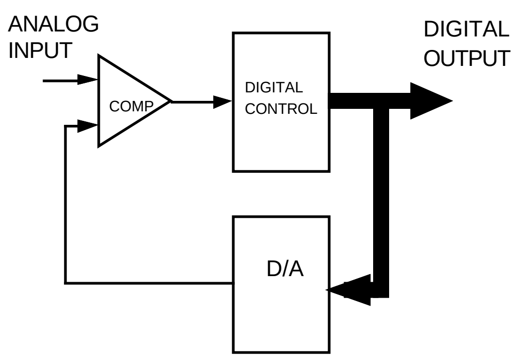

Figure 2. Basic building blocks for direct conversion.

With *indirect* conversion, the analog signal is converted into a time or frequency domain signal that is then measured by the digital logic and converted to a binary value.

The following A/D conversion schemes are discussed in the following sections:

- Dual-Slope integrating converter
- Counting converter
- Successive approximation converter
- Voltage-to-frequency converter

The actual implementation of these techniques can vary significantly from one system to another depending on the requirements of the system. Such factors as conversion speed, accuracy, resolution, linearity and cost all affect the overall design of a particular A/D conversion sub-system.

Dual-Slope Integrating Converters

A very popular form of the indirect converter, the dual-slope integrating converter incorporates an analog integrator. The input analog signal is first integrated over a fixed period of time, T1. The integrator's input is then switched to a reference voltage, V_{ref1}, where it is allowed to integrate down while a digital counter is incrementing. The counter is stopped when the integrator's output reaches a preset voltage, V_{ref2} (T2 period). The contents of the counter then represent the converted digital value. Figure 3 shows the basic building blocks of a dual-slope integrating converter.

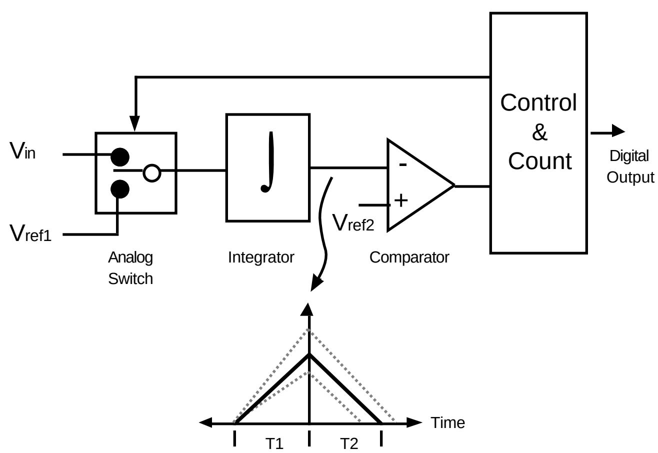

Figure 3. Basic building blocks for integrating converter

The dual-slope integrating converter has several important advantages. Because of the complementary nature of the two opposing slopes, the conversion accuracy is independent of the accuracies of both the clock frequency and the integrator capacitor. Also, due to the inherent nature of the built-in integrator, the output digital value represents an averaging of the analog input signal over the integration period. This makes the converter highly immune to input noise.

A disadvantage of the dual-slope integration is that the required integration period places a restriction on the maximum conversion speed. Dual-slope integrating converters are thus generally used for slower application such as environmental monitoring of temperature or humidity.

The Neuron Chip includes a dual-slope integrating converter as one of its built-in I/O objects called dualslope input. The Neuron Chip essentially replaces the control and counter block shown in figure 3. An example of the use of the dualslope input object will be presented later.

Counting Converters

This is the simplest form of the direct conversion technique. The implementation flexibility of this scheme allows for wide design variations, from an entirely hardware-based design to a combined hardware-software implementation. This allows the use of already available resources within the Neuron Chipin order to reduce overall cost and complexity.

There are two basic types of counting converters: stair-step and tracking. In the stair-step scheme, the count and control functional block shown in figure 2 is an N-bit up-counter whose counting operation is controlled by the comparator output.

At the start of the conversion cycle, the counter simply starts counting up from zero until the output of the D/A converter equals the input analog signal. At that point, the counter stops incrementing. The N-bit binary number in the counter represents the digital output corresponding to the analog input signal. The counter is then reset and the entire sequence is repeated for another conversion.

The conversion speed of this technique is relatively low due to the inherently slow counting process required. For an N-bit converter, the worst-case (full-scale) conversion time is 2^N clock cycles.

The counting operation can also be performed by software. Although logically equivalent to the all-hardware implementation, this scheme allows the use of already available resources (microprocessor or microcontroller) in order to reduce the amount of external logic. The use of software also permits design modification (e.g., changes in the conversion algorithm) at a much lower cost.

The basic structure for a software-oriented counting-type A/D converter is shown in figure 4.

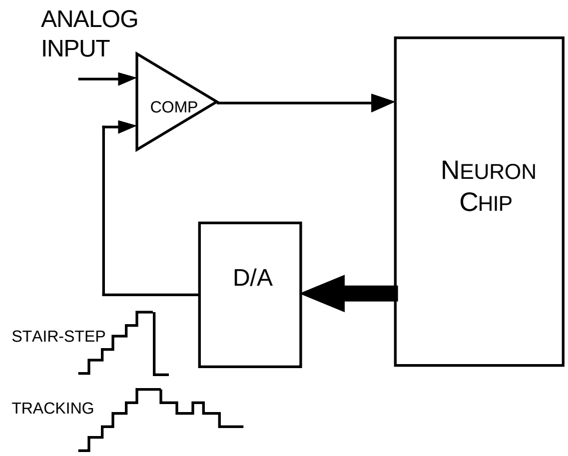

Figure 4. Software-driven counting-type A/D converter. The processor replaces the digital control block shown in Figure. 2.

The average conversion speed of the stair-step counter converter may be increased by using an up-down counter. In a tracking or servo-type converter, the counter is allowed to count in both directions and track the input analog voltage. This translates to shorter conversion time for small changes in the analog input. For large changes, however, the conversion time approaches that of the original stair-step converter. The tracking counter converter can also be implemented in software (figure 3) by modifying the software to behave accordingly.

Counting converters (stair-step or tracking) are suitable in applications where conversion speed is of small concern relative to the cost and simplicity of the design.

Successive Approximation Converters

This direct-type conversion scheme is similar to the counting type A/D converter in all but the count and control section, as shown in figure 5.

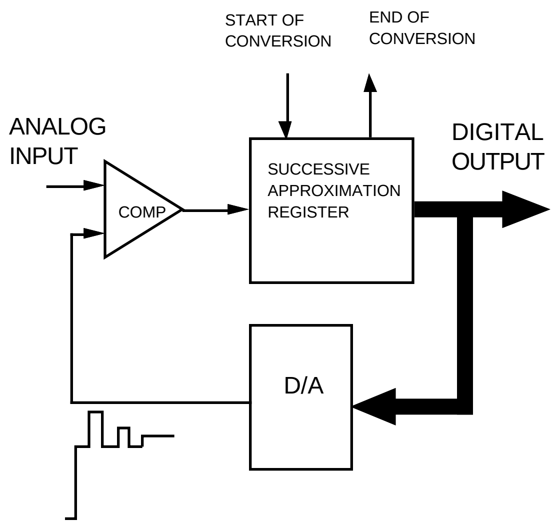

Figure 5. Basic building blocks for a successive approximation-type A/D converter.

The successive approximation register bypasses the long and slow process of incrementing the count sequentially. This is accomplished by toggling each of the N bits of the digital output one at a time, starting with the MSB, while monitoring the output of the comparator. With this technique, only N cycles are needed for a conversion.

The successive approximation converter has gained popularity due to its relatively balanced attributes (speed, complexity and cost). A number of off-the-shelf converters on the market today are of the successive approximation type. Some of the following design examples will use such ICs.

The circuit shown in figure 4 can also be used to implement a successive approximation converter. The only change is a software modification that implements the successive approximation algorithm instead of the counting algorithm in the microprocessor or microcontroller.

Voltage-to-Frequency Converters

This indirect conversion technique transforms the analog signal into the frequency domain. The frequency is then measured with a separate circuit or by a software routine. A block diagram is shown in figure 6.

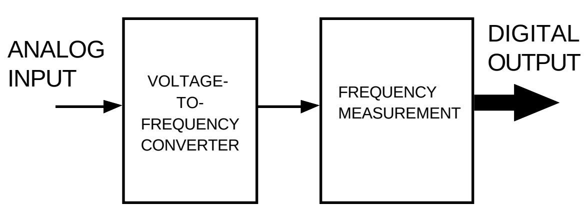

Figure 6. Voltage-to-frequency conversion.

The interface between the voltage-to-frequency (V/F) converter block and the frequency measurement block is a two-wire connection that carries a waveform with a frequency proportional to the amplitude of the input analog signal.

The V/F conversion technique is therefore useful in cases where the analog sensor is physically distanced from the rest of the system. In such a setup, the V/F converter is situated next to the source of the analog signal. A connecting pair of wires carries the frequency output to the main system for further processing. Note that the analog signal no longer has to travel for a long distance, thus avoiding signal degradation and interference.

A variation on the V/F method is the pulse-width conversion method. In this technique, the amplitude of the analog signal determines the output pulse width of a one-shot circuit. The pulse width, as with the frequency in the V/F case, is then measured and appropriately converted to a digital number.

As with any other conversion scheme, there is a balanced tradeoff between speed of conversion and resolution. A unique characteristic of the V/F conversion technique, however, is the high degree of control the designer has over this balance.

By changing the conversion scale of the V/F converter block shown in figure 6, for example, one can easily increase or decrease the resolution while decreasing or increasing the conversion speed.

The Neuron Chip, with its built-in timer/counter, easily lends itself to the above time measurements. The input objects than can be used for time/period measurements are ontime, period, and pulsecount, all having 16-bit resolution.

Other A/D Conversion Techniques

There are a number of other analog conversion techniques that we have not discussed here. This should by no means reflect on their relative importance in the A/D conversion arena. The scope of the conversion techniques discussed has been limited to the ones that directly relate to the Neuron Chip interface and capitalize on its features.

Conversion techniques such as flash, half-flash, and delta-sigma are addressed in many texts, some of which are noted in the reference section at the end of this engineering bulletin.

Sample/Track and Hold

In the previous conversion techniques, a basic assumption was implied about the rate of change of the input analog signal.

Generally, input signal changes of greater than 1/2 LSB during any of the conversion processes results in a misleading digital output with the exception of the V/F converter and dual-slope integrating converter which inherently averages the input signal over the integration period. Therefore, it is assumed that the input is constant, or that the change is so small that it is not recognized by the A/D converter (e.g., outdoor light intensity or temperature).

For a time-varying analog signal, there is a need for circuitry that stabilizes the input to the A/D converter while the conversion process is taking place. The simplest approach is to use a low-pass filter that discriminates against higher frequencies (slew rates) that fall outside the converter's sampling rate. Another option would be to use a sample-and-hold (S/H) circuit.

A S/H block, as shown in figure 1, takes a snap-shot of the input signal when instructed to do so, and holds that value until the next sample.

A track-and-hold (T/H) circuit, on the other hand, spends most of its time tracking the input voltage and switches to the hold mode for brief periods of time when instructed to do so.

A wide selection of S/H and T/H ICs exist on the market that enable the designer to bypass the design of this functional block in all but the most demanding cases. Most S/H ICs are also capable of performing in the T/H mode.

Analog Multiplexers

Analog multiplexers, as shown in figure 1, permit the use of only a single A/D converter (and its associated S/H) for acquiring analog signals from multiple sources. They perform in the analog domain what digital multiplexers do in the digital domain.

An analog multiplexer has the advantage of fast switching time without the ‘bounce’ typically associated with mechanical relays. However, since semiconductor switches are used in these devices, the ON resistance might affect the measurement in some design situations.

As with the S/H case, a variety of off-the-shelf analog multiplexer ICs are available on the market to aid the designer.

Voltage References

In any A/D conversion design, there is a need for a point of reference against which the input signal must be compared.

When measuring physical quantities, the analog input is either an absolute signal (e.g., voltage or current) or it is in the form of a varying ratio. In the case of an absolute signal, the converter must be able to measure absolute levels of input.

For example, a thermocouple, which has a 0 to 0.25 volt output range, must be used with an A/D converter that can recognize an absolute voltage of 0.125 as the half-scale value. This is accomplished by providing an accurate reference to the A/D converter.

In the case of a varying ratio, absolute accuracy is not important. This is referred to as ratiometric measurement since the measured analog signal is dependent on a ratio provided by the transducer and not the voltage driving the transducer.

Both referencing schemes are used today as they each address a different need. An absolute referencing technique must be used where the analog sensing device produces an absolute voltage. Many such sensors exist today. Semiconductor transducers that measure such physical quantities as temperature, force and acceleration produce an absolute voltage relative to the measured quantity.

Use of the absolute referencing design, however, requires an accurate and stable reference voltage circuit which might be a disadvantage in terms of cost and complexity. The ratiometric approach eliminates this need while providing highly accurate measurements. Since both the measurement transducer and the A/D converter use the same reference voltage, voltage changes are observed by both and are therefore cancelled out.

The use of this scheme requires, however, a transducer capable of providing a variable impedance or ratio as an output, much like a center-tapped potentiometer.

Design Examples

The following is a series of actual design examples using the Neuron Chip. All circuits and their accompanying software have been tested to validate their functionality.

Several off-the-shelf ICs have been used with the designs in order to provide a good sampling of the available options. In addition, an A/D converter circuit discussed uses the internal functions of the Neuron Chip with minimal additional logic to accomplish its task.

Using an A/D converter IC with a parallel interface requires more I/O pins than a serial interface. In addition, the interface is further complicated by any converter with a resolution greater than eight bits. For this reason, the off-the-shelf A/D converter ICs used in the following examples are serial.

In the following circuits, as with any A/D circuit in general, careful attention must be given to potential noise and error sources such as ground loops, offset voltages and currents, and layout techniques. This is especially important with higher resolution converters.

Example 1

As mentioned before, the Neuron Chip supports a dual-slope integrating converter through its dualslope input object. One of the internal timer/counters of the Neuron Chip is used to control and measure the integration process. This permits a conversion resolution of up to 16 bits at a integration period of 13.11ms (10MHz input clock). Faster conversion rates can be attained at the expense of bit resolution.

Figure 7 is a typical converter circuit using the dualslope object.

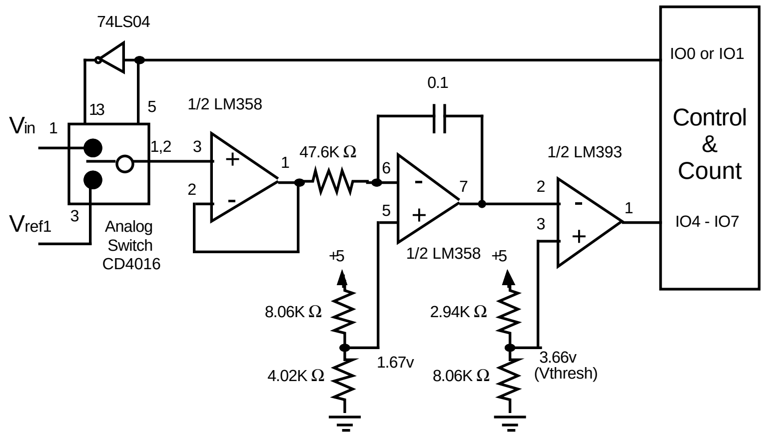

Figure 7. An example of a dual slope integrating A/D using the Neuron Chip's built in dualslope input object

For the dualslope input object the Neuron Chip's timer counter is set up as both an input and output device. The output signal typically is used to select the input source for the integrator: either the unknown analog input voltage, or a known reference voltage. The input signal to the timer counter is typically the output of a comparator which is comparing the integrator output to a reference voltage or zero voltage reference.

Declaring the dualslope input object will configure a timer counter to perform two important functions: count only when the input signal is active, and latch the count on the falling edge of the input signal. The maximum value for either of the integration periods can be controlled by the `clock (n)` keyword in this object declaration. See the *Neuron C Reference Guide* for the ranges provided by the various clock settings. The `invert` keyword may also be used here if the input signal is active low rather than active high.

The conversion process must be started by the application, through the use of the `io_in_request()` function. The end of the conversion is detected by the firmware by monitoring both the timer/counter control output and input signals. Therefore the application does not need to wait around during the conversion process, its completion is detected as an event through an `io_update_occurs()` `when` clause.

Figure 8 illustrates the relationships between the timer/counter input and output signals, and the integrator output. Three different input voltage integration slopes are shown for comparison.

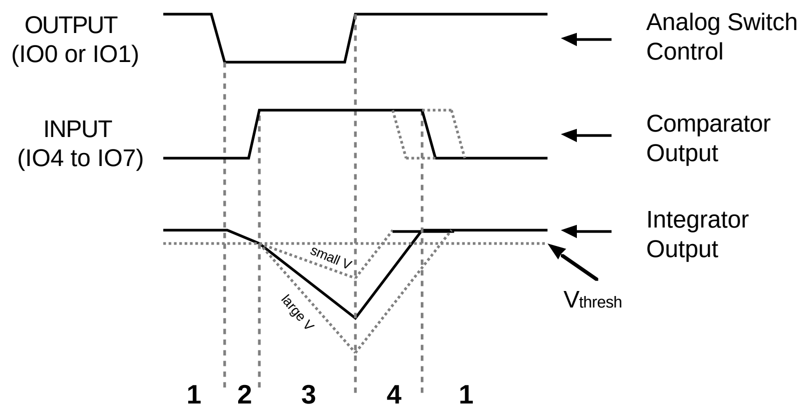

Figure 8. Various states of the dualslope converter

State 1 is the idle state. The timer/counter is not counting since the input signal from the comparator is low. The integrator input is selected to the reference voltage.

State 2 is entered by calling the `io_in_request()` function for the dualslope input object. This step performs two tasks: it loads the timer/counter with the value (defined by the `control_value` passed by the `io_in_request()` call) that will produce the first integration period, which is typically a constant period, and, it activates the timer/counter output signal, which in turn selects the V_{in} (unknown analog input) signal as the input to the integrator. Note that until the integrator crosses the comparator threshold, which in turn activates the timer/counter input signal, the timer/counter is not counting. This ensures that the first integration period starts at this threshold point rather than at the start of State 2.

State 3 starts when the timer/counter input signal goes active and runs for the length of the first (fixed) integration period.

State 4 starts when the timer/counter reaches terminal count. At this point the timer/counter output toggles, selecting the V_{ref1} input as the input to the integrator. This also re-loads the timer/counter with the value defined by `control_value`. This means that the timer/counter will count from this value rather than zero, and implies that the second integration period will not be longer than the first. This is usually the case with this type of converter. State 4 ends when the integrator output crosses the comparator threshold in the other direction. The timer/counter input signal goes inactive, the current count is latched, and the timer/counter stops counting as State 1 is entered. At this point the timer/counter latch is read and stored into the `input_value` variable, and may also be retrieved through an `io_in()` function call. This value reflects the second integration period (State 4 period), which reflects the unknown V_{in} value as compared to the V_{ref1} value.

The value stored in `input_value` will always be biased by the two's complement of the value passed to the `io_in_request()`, and will lie between this value and the overflow value (0xFFFF).

In this example a 5MHz node will control a dual slope converter with a 24ms integration period. The conversion will occur every 500ms. The dedicated timer counter is used with a clock (0) setting which yields a 400ns resolution at 5MHz. IO_4 is the input signal to the timer/counter, and IO_1 is the output signal from the timer/counter. An `io_in_request()` count value of 60,000 will produce a 24ms integration period and provide a resolution of just under 16 bits.

Note that, in addition to decreasing the resolution, the integration period could also be reduced by reducing the overall voltage span of the integrator output. This can be accomplished by reducing the level of the comparator reference voltage. This however can lower the system signal-to-noise ratio and must be considered carefully.

An application timer will be set up to start the process periodically. Another application timer is used to detect an underrange condition. This condition can occur if the second integration period goes on forever due to an underrange input voltage.

```
// A/D conversion using the dualslope input
// of the Neuron Chip.
// Perform a measurement every 500ms

mtimer repeating go_time;
mtimer fail_time;
unsigned long raw_ds;
IO_4 input dualslope ded clock (0) dsad_1;

when(reset) {
   go_time = 500;                // Convert every 500ms.
}

//Conversion is started by this event

when(timer_expires(go_time)) {   io_in_request(dsad_1, 60000UL);
   fail_time = 80;      // Underrange if 80ms passes.
}
//End of conversion is detected by this event

when(io_update_occurs(dsad_1)) {
```

```
   raw_ds = input_value + 60000UL;
   fail_time = 0;    // Stop the timer.
   // raw_ds may now be scaled and used.
}
//Error condition is handled by this event
when(timer_expires(fail_time)) {
   // This is an error condition usually caused by an input voltage
   // that is too low - the comparator output never went active.
}
```

Example 2

Figure 9 illustrates a typical connection of the Neuron Chip to the Motorola MC145053 single-chip A/D converter IC for performing ratiometric measurements.

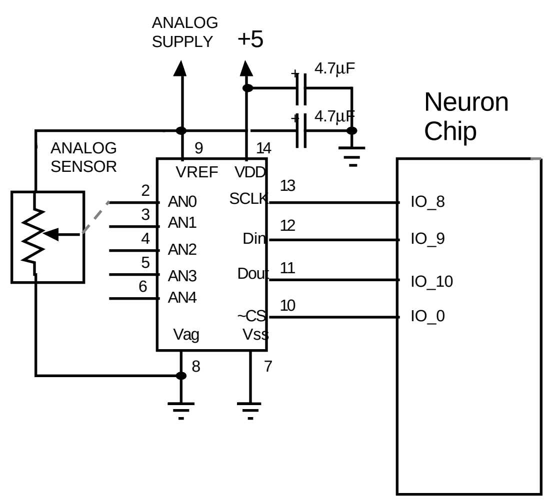

Figure 9. An example of an off-the-shelf A/D solution with the Neuron Chip using the Motorola MC145053.

This is an example of a 10-bit, multi-channel, full-duplex, serial A/D IC. The address information is clocked into the IC at the same time the converted digital output is clocked out.

Using the Neuron Chip's neurowire I/O object, also a full-duplex interface, we can simultaneously shift out the new address and shift in the converted data from the previous address.

An internal S/H function is provided by the MC145053. In addition, three test addresses are available for verification of the operation of the chip. Refer to the MC145053 data sheet for specific feature and timing information.

The Motorola MC145051 is identical to the MC145053 except that it provides 11 analog input channels in a 20-pin package.

The following is an example of typical Neuron C code for use with the circuit of figure 9.

```
// A/D conversion function for the
// Motorola MC145053 (and MC145051).
// The function analog_to_digital(analog_addr) returns a
// 10-bit number corresponding to the analog input with
// address of analog_addr from previous call.

///////////////////I/O declarations/////////////////////
IO_8 neurowire master select (IO_1) ADC_IO;        //A/D chip interface
IO_1 output bit ADC_CS = 1;                        //running at 20Kbaud

///////////////////Function declaration////////////////
unsigned long analog_to_digital(unsigned long analog_addr); //function
                                                            //protocol
///////////////////Function definition////////////////
unsigned long analog_to_digital(unsigned long analog_addr) {
static unsigned long adc_info;      //raw data
unsigned long digital_out;          //returned digital data
   adc_info = analog_addr<<12;      //shift new addr out, and
   io_in(ADC_IO, &adc_info, 10);    //get converted data for previous address
   digital_out = ((adc_info>>6) & 0x3fc) | (adc_info & 0x03);
                                    //shuffle bits for correct format
   return digital_out;
}
```

The neurowire interface is operating at 20kbps (default). Alternatively, it could be configured for operation at 10 or 1kbps, if using an A/D converter IC that cannot work at such a high speed.

The select pin for the neurowire is chosen to be the IO_1 pin in this example and must be given an initial value of logic 1, as required by the IC.

The address that is clocked into the chip must reside in the four least significant bits of `analog_addr`. Also, the data clocked out of the IC, `adc_info`, needs to be shifted due to the fact that the MSB is shifted out first, in blocks of eight bits. For more information on the operation of the neurowire I/O object refer to the *Neuron C Reference Guide*.

The conversion rate is dependent on the neurowire bit rate selected, the particular mode the MC145053 is used in, and the actual program itself. Using the above program (20kbps, 16-bit mode), a complete conversion can be performed in about 1ms.

Example 3

Figure 10 is an example of multiple A/D ICs interfacing to the Neuron Chip. The Linear Technology LTC1095 is a multi-channel, 10-bit successive approximation, half-duplex serial A/D IC with additional features such as S/H, voltage reference, single-ended or differential analog input modes, and unipolar or bipolar operation, all conveniently integrated into a single IC.

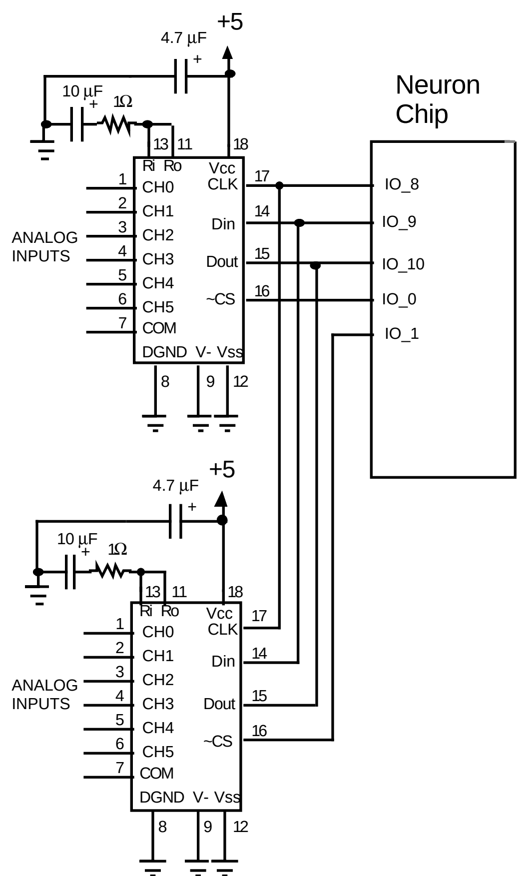

Figure 10. Multiple A/D interface using Linear Technology's LTC1095.

The circuit in figure 10 uses the absolute referencing technique discussed earlier. The internal, 5 volt, reference voltage generator of the IC provides an accurate source for scaling the input analog voltages. In the unipolar mode of operation, this translates to an input range of 0 to 5 volts. In the bipolar mode, the range is increased to -5 to 5 volts. Both modes assume a direct connect of the LTC1095 reference output pin to the reference input pin.

The reference input of the LTC1095 can be connected to other reference voltages to accommodate any input signal range. The availability of an input reference pin permits ratiometric referencing.

In addition to the analog channel address, the LTC1095 requires other configuration information. The format of the 7-bit configuration word is as follows:

START	SGL/ ~DIFF	ODD/ ~SIGN	SEL 1	SEL 0	UNI	MSBF

Each of the above fields permits the user to configure the different features of the IC for different applications. The above configuration bits are defined as follows:

START - Initiates data transfer

SGL/~DIFF - Inputs referenced to COM, or differential pairs

ODD/~SIGN - Select the single-ended input, or polarity of differential pair

SEL0, SEL1 - Channel selection

UNI - Unipolar or bipolar operation

MSBF - MSB out on data output from the converter

Additional information on the timing and functionality is contained in the LTC1095 data sheet.

Since this IC uses a half-duplex communication protocol, the address clock-in and the data clock-out cycles do not occur simultaneously. This implies that the data can be accessed immediately after the address has been clocked into the IC.

The address clock-in and data clock-out are part of a single operation, meaning that the Chip Select (~CS) line must be held active during the full sequence.

A neurowire I/O function call normally activates the designated select line at the beginning of the operation and deactivates it at the end. Therefore, unlike the previous example, we cannot use the full-duplex neurowire function.

The bitshift I/O object is used in conjunction with the neurowire I/O object in this example. The address is clocked-out to the LTC1095 first using the bitshift function. This is immediately followed by a neurowire function call that clocks-in the converted data from the previous address in to the Neuron Chip. The neurowire is used, therefore, as a half-duplex interface.

The ~CS line is held active (low) throughout this entire sequence of events by overlaying the select line of the neurowire with a bit output object. This allows for separate control of that line by both the neurowire object and the application program.

The following is a Neuron C program that can be used with the circuit of figure 10.

```
// A/D conversion function
// for the Linear Technology LTC1095.
// The function analog_to_digital(analog_addr) returns a
// 10-bit number corresponding to the analog input with
// address of analog_addr, from the ADC designated by adc_num.

///////////////////I/O declarations/////////////////////
IO_0 output bit ADC_CS_1 = 1;
IO_1 output bit ADC_CS_2 = 1;
IO_8 neurowire master select (IO_0) ADC_IO_1;   //A/D chip interface
IO_8 neurowire master select (IO_1) ADC_IO_2;   //A/D chip interface
IO_8 output bitshift numbits (8) clockedge (+)  //A/D chip interface
   ADC_group_control;

///////////////////Function declaration////////////////
unsigned long analog_to_digital(int adc_num, unsigned short analog_addr);

///////////////////Function definition////////////////
unsigned long analog_to_digital(int adc_num, unsigned short analog_addr) {
static unsigned long adc_data;         //raw data from A/D chip
unsigned long digital_out;             //returned digital data
static unsigned short addr;            //analog address/config
                                       //info to A/D
   addr = analog_addr;
   if (adc_num==1)io_out(ADC_CS_1, 0); //Activate first ADC
   if (adc_num==2)io_out(ADC_CS_2, 0); //Activate second ADC
   io_out(ADC_group_control, addr);    //send addr info to A/D
   if (adc_num==1) io_in(ADC_IO_1, &adc_data, 16); //get converted data
   if (adc_num==2) io_in(ADC_IO_2, &adc_data, 16); //get converted data
   digital_out = adc_data >> 6;        //right justify
   return digital_out;
}
```

The function `analog_to_digital(adc_num, analog_addr)` returns the converted data for the specified converter, `adc_num`, and the given configuration information, `analog_addr`.

One completed A/D conversion cycle using the above program requires 8 bitshift clock cycles (15kbps), 16 neurowire clock cycles (20kbps), and the latency associated with executing the code to perform the I/O functions. For a Neuron Chip running at 10MHz, the total conversion time is approximately 1.5 ms.

It is important to follow the rules outlined in the LTC1095 data sheet for implementing a ground plane. Proper bypassing, good layout techniques, and minimization of noise on the reference input are also important for reducing errors to a minimum.

Example 4

This example makes use of the available resources within the Neuron Chip to implement a stair-step counting type A/D converter suitable for relatively low conversion rates and resolution.

The design in figure 11 is perhaps the simplest of all the examples covered in this document in terms of hardware requirements external to the Neuron Chip. The counter and its associated logic needed for this type of conversion are implemented in software using Neuron C. To simplify the circuitry even further, the D/A converter is reduced to a simple passive integrator driven by the pulsewidth output object of the Neuron Chip.

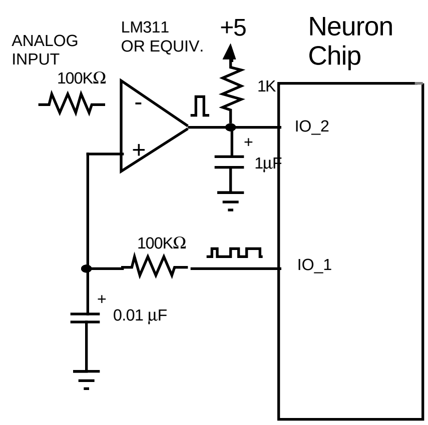

Figure 11. A counting type converter with some help from the Neuron Chip.

The pulsewidth output object of the Neuron Chip produces a continuous train of pulses. The frequency of the output is set during the declaration of the output object and is therefore set at compile-time to one of the eight available frequencies, from 152.6Hz to 19.53kHz. The duty cycle of the output, however, can be varied under program control. This provides a means by which to produce an analog voltage using a simple resistor-capacitor network.

The resolution of the pulsewidth output object can be set at either 8 or 16 bits. Therefore, it is theoretically possible to use the circuit of figure 11 for a converter with a resolution of up to 16 bits. There are, however, practical limitations in implementing a converter with such high resolution.

The highest output frequency at the 16-bit resolution is around 76Hz. At this frequency, the time constant of the RC network shown in figure 11 must be increased to accommodate the increased output period. This, in turn, must be accompanied by extra delays incorporated in the program to allow voltages to settle. This translates to longer conversion times that may be unacceptable in certain applications.

In addition, without careful attention to noise suppression and ground looping effects, such a converter circuit would produce erroneous results at higher resolutions.

Very good stability can be achieved considering the simplicity of the circuit in figure 11. This is due to the fact that the pulse-width output waveforms of the Neuron Chip are derived from a stable crystal time-base.

The resistor and capacitor values in figure 10 are chosen to provide a balance between circuit speed and ripple rejection, given a pulse-width output resolution of 8 bits. For large time constants, the Neuron Chip must wait for the capacitor to charge up to the new value for every update of the duty cycle. For small time constants, the voltage at the non-inverting input of the comparator would not be stable between duty cycle transitions. The values shown were used for a pulse-width frequency of 9.77kHz.

Careful attention must be given to proper grounding, layout, and component selections in order to reduce conversion error and increase stability. This is especially important when working with higher resolution converters. For example, an improperly biased comparator could result in missed codes at the beginning or at the end of the conversion range.

The design in figure 11 can be adapted for a tracking type counting converter. This is achieved by modifying the software to count in both directions. That is, the output of the comparator would be used by the Neuron C program to determine the direction of the next count change.

The program below is an example of typical code that can be used with the circuit of figure 11 to implement an 8-bit A/D converter.

```
// counting type ADC function using
// the Neuron Chip's internal resources
#include <control.h>

///////////////I/O Declarations//////////////////
IO_2 input bit match;                        //comparator input
IO_1 output pulsewidth clock (1) DAC_out = 0;//PWM output, 8-bit resolution

///////////////Function prototype/////////////
unsigned short analog_to_digital(void);

///////////////Function definition////////////
unsigned short analog_to_digital(void) {
unsigned int digital_out =0;        //initialize to zero
   io_out(DAC_out, 0);              //reset output
   delay(4000);                     //wait for cap to discharge
   digital_out = 0;
   while (io_in(match)==0) {        //ramp up the DAC output
      io_out(DAC_out, digital_out);
      digital_out++;                //until it equals the input signal
      if (digital_out == 255) break;   //ceiling is 255
      watchdog_update();            //tickle watchdog timer
   }
   return digital_out;
}
```

Example 5

This example is a variation of example 4. The circuit used in the previous example (figure 11) can be used to implement another conversion technique by using a different software algorithm.

The slow conversion of the stair-step is improved in this example by using the successive approximation algorithm. The program below is an implementation for such an algorithm, written in Neuron C, that implements an 8-bit converter.

```
// Successive approximation type ADC function
// using Neuron Chip's internal resources
#include <control.h>

/////////////////////////Declarations////////////////////
IO_2 input bit match;                               //comparator input
IO_1 output pulsewidth clock (1) DAC_out = 0;   //PWM output, 8-bit resolution

//////////////////////Function Prototype////////////////
unsigned short analog_to_digital(void);

////////////////////////Function Definition/////////////
unsigned short analog_to_digital(void) {
unsigned int digital_out;
unsigned int bit_num;
digital_out=0;
for (bit_num =0x80; bit_num!=0; bit_num >>=1) { //setup for //8 bit conv.
   io_out(DAC_out, digital_out+bit_num);        //increment output
                                                //by bit weight
   delay (1000);                                //wait for RC to settle
   if (io_in(match)==0) digital_out+=bit_num;   //set new output
   watchdog_update();
   }
   return digital_out;
}
```

Note that each conversion now takes a constant number of cycles equal to the number of bits desired in the digital output.

As previously mentioned in example 4, this program could also be implemented using the 16-bit feature of the pulsewidth output object. The same limitation, however, discussed in the previous example still applies.

As always, the circuit should be designed with close attention to biasing and grounding. Proper comparator offset is especially important as it can be the major source of error in this design.

Example 6

This example is an implementation of the voltage-to-frequency conversion technique, providing the simplest interface to the Neuron Chip compared to all previous examples.

The 555 timer is used to produce a square-wave whose frequency depends on an analog voltage. Figure 12 shows the circuit and its connection to the Neuron Chip.

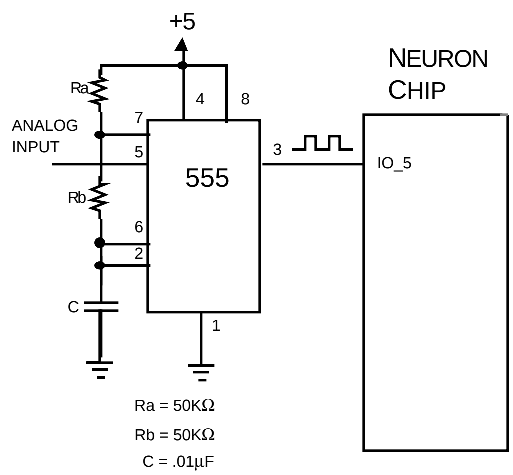

Figure 12. Voltage-to-frequency conversion using the 555 timer.

As shown in figure 12, only one pin is needed for connection of the 555 to the Neuron Chip. This design, therefore, is well suited for applications where Neuron Chip I/O resources are at a high demand.

In addition, in applications where the 555 timer is not situated near the Neuron Chip, this design provides for the minimum amount of interface wiring.

The 555 timer IC in figure 12 is wired as an oscillator with an output frequency that is dependent on the input voltage level on pin 5, in addition to Ra, Rb, and C.

The 555 used in the above configuration does not provide a linear relationship between voltage and frequency, desirable in applications requiring relative accuracy. Also, since the circuit is not a closed-loop system, it is more susceptible to output variations due to temperature, humidity, and long-term drift.

The circuit in figure 12 does however provide a simple and inexpensive solution for providing relative (non-absolute) conversions within a relatively small input voltage range.

For applications requiring higher precision and accuracy, an off-the-shelf voltage-to-frequency converter, such as LM131, may be used. This IC is specifically designed for A/D conversion applications and provides very good stability and a wide range of full scale frequencies in a small 8-pin DIP package.

Regardless of the circuit used to convert an analog voltage to frequency, the software in the Neuron Chip must be able to extract a meaningful value from the incoming waveform.

The Neuron Chip provides several ways for accomplishing this task. The ontime, period, and pulsecount input objects of the Neuron Chip may be used to convert an input square wave of a varying frequency into a digital number. All three methods provide a 16-bit resolution.

The ontime input object measures the duration of either the high or low (user selectable) section of one input waveform cycle. The conversion time is therefore dependent on the absolute frequency range and the amplitude of the input voltage. A drawback of the ontime method is the instability of the output due to cycle-to cycle variations in waveform timing.

The ontime input function can also be used in a voltage-controlled one shot conversion system, where the output pulse-width of the one shot is a function of the input voltage.

The period input object performs the same measurement as the ontime input object, except the entire cycle is measured. All attributes of the ontime object scheme apply to the period object scheme. One advantage is that, since a larger piece of the waveform is now being measured, the contribution of the errors due to cycle-to-cycle variations and input range limitations are less pronounced.

The pulsecount input object provides an alternate solution by measuring the number of input edges within a 0.8388608 second time frame. This implies a constant conversion time.

The major advantage in using the pulsecount input function block is the inherent averaging performed by the function itself. Since the actual number of pulses are counted within a given time period, the minor cycle-to-cycle time variations will be averaged over many cycles and would therefore have a much smaller impact on the overall conversion result.

It should also be noted that the pulsecount object, unlike ontime and period, provides an inverse relationship between input voltage and the digital number. An increase in the input voltage causes a decrease in the frequency which translates to a lower count value returned by the pulsecount function.

The following is an example of typical code for use with the circuit in figure 12. The pulsecount input object is used to implement a 16-bit converter.

```
// A/D conversion using a 555 timer
// as a voltage-to-frequency converter
#include <control.h>

//////////////////////// I/O Declaration ///////////////////
IO_5 input pulsecount analog_input;

////////////////////////Function declaration////////////////
unsigned long analog_to_digital(void);

////////////////////////Function Definition////////////////
unsigned long analog_to_digital(void) {
unsigned long digital_out;
   digital_out = io_in(analog_input);
   return digital_out;
}
```

Performance Issues

Several factors must be taken into account when dealing with real-time, asynchronous, analog signals in a system. These factors become especially important in a distributed network, where the interaction of one node with the others directly affects the network performance and therefore the overall system functionality.

In a distributed control environment, a typical node might convert an analog signal to a digital number, either for direct measurement of the level of the signal, or for calculating another parameter that depends on the analog signal.

In either case, if the resultant digital number is placed on the network for transmission to one or more nodes, overall network traffic is affected, depending on how and when the information is passed, and on the behavior of the analog signal itself.

For example, if the converted digital number is passed from one node to another each time there is a new analog value, the number of packets transferred is directly proportional to how often the sending node performs a conversion.

It is generally not desirable to allow network traffic to be controlled so closely by an external event. This is specially true in the case of an A/D converter where the digital output can change as a result of noise or quantization error.

Therefore, in order to minimize the amount of redundant network traffic and to improve overall system response time, one or more of the following techniques should be used.

1) Using polled network variables. This would cause a packet transfer only when the receiving node requests the sending node to do so.

2) Making A/D conversions less frequently. By optimizing the rate of conversion, redundant packet transmission is reduced. This would also free the application processor for other tasks.

3) Introducing hysteresis by sending updates over the network only when the change in the converted digital number is larger than a specified value. See the *LONMARK™ Application Layer Guidelines* for standard definitions for hysterisis values.

4) As a variation on (2) above, time-averaging the digital values between update intervals.

A/D Converter Sources

The following is a partial listing of serial A/D converter ICs available on the market. The ICs used in the previous examples are included.

PART NUMBER	BITS	CONV. SPEED	PKG SIZE	# OF CHN	S/H	REF	COMMENTS
Analog Devices							
AD575	10	30 µs	14 pin DIP	1	N	Y	Dual supply, unipolar or bipolar
Linear Technology							
LTC1091	10	20 µs	8-pin DIP	2	Y	N	Vcc=Vref
LTC1092	10	20 µs	8-pin DIP	1	Y	N	
LTC1094	10	20 µs	20-pin DIP	8	Y	N	Unipolar/bipolar
LTC1095	10	20 µs	18-pin DIP	6	Y	Y	Vcc=Vref
LTC1291	12	13 µs	8-pin DIP	2	Y	N	
LTC1292	12	13 µs	8-pin DIP	1	Y	N	
LTC1294	12	13 µs	20-pin DIP	8	Y	N	
National							
AD0831	8	32 µs	8-pin DIP	1	N	N	
AD0832	8	32 µs	8-pin DIP	2	N	N	
AD0834	8	32 µs	14-pin DIP	4	N	N	
AD0838	8	32 µs	20-pin DIP	8	N	N	
Motorola							
MC145051	10	88 µS	20-pin DIP	11	Y	N	
MC145053	10	88 µS	14-pin DIP	5	Y	N	
Maxim Integrated Products							
MAX170	12	5 µs	8-pin DIP	1	N	Y	Dual supp, uni/bi
MAX171	12	5 µs	8-pin DIP	1	N	Y	Opto-isolated
MAX190	12	13 µs	24-pin DIP	1	Y	Y	

References

1) Analog Devices, Analog-Digital Conversion Handbook, Prentice-Hall, 1986.

2) Analog Devices, Data Conversion Products Data Book, 1989/90.

3) Linear Technology, 1990 Linear Data Book.

4) Maxim Integrated Products, Integrated Circuits Data Book, 1990.

5) Motorola, CMOS ASIC ICs, Q4/90.

6) National Semiconductor, Data Acquisition Linear Devices Handbook, 1989.

Part Number 005-0019-01 Rev. C

Echelon Corporation
4015 Miranda Avenue
Palo Alto, CA 94304
Telephone (415) 855-7400
Fax (415) 856-6153

Echelon Europe Ltd
Elsinore House
77 Fulham Palace Road
London W6 8JA
England
Telephone +44-81-563-7077
Fax +44-81-563-7055

Echelon Japan K.K.
Kamino Shoji Bldg. 8F
25-13 Higashi-Gotanda 1-chome
Shinagawa-ku, Tokyo 141
Telephone (03) 3440-7781
Fax (03) 3440-7782