A Zytek Communications Corporation White Paper

Input / Output Expansion Without Control Lines

Stephen Melvin, Ph.D.

November 4, 2011

Input / Output Expansion Without Control Lines

Abstract

Expansion of input and output lines in an embedded system is sometimes complicated by the unavailability of any additional control or select lines. This paper discusses a technique to expand the input and/or output of an embedded system without requiring any control signals or requiring that there be any unused states on existing signals. The key idea is to utilize transitions between states of existing output lines to generate new control signals and then reprogram the embedded system such that transitions on those output lines are controlled in a specific way.

Introduction

Embedded systems typically incorporate a host microprocessor or microcontroller coupled to peripherals devices. Typically, signals coupled between the microcontroller and the peripheral are used for the input of data from such peripherals and for the output of data to such peripherals. These data signals can be directly wired to the microcontroller or there can be intervening buffers or registers. Typically certain of these signals are control signals, such as enable signals or strobe signals, which indicate to the peripheral when to perform data input or output respectively. Alternatively, control signals can be implemented in the form of select lines, which are used to indicate to the peripheral how to interpret other signals.

A problem that arises with embedded systems is the need to expand the I/O beyond the number for which it was originally designed. For example, it may be necessary to add an additional eight output signals to an embedded system that was only designed with the consideration of handling 16 output signals. Unless there are unused control signals that can be utilized for such an expansion, significant changes may be required, including substantial redesign and additional wires. In the case of select lines, expansion is sometimes simplified if there are unused states, but often all states have been defined and are utilized by existing peripherals. Again this means that a significant redesign effort may be required.

This paper discusses a mechanism to expand the I/O capabilities of an embedded system without requiring any additional control signals and without requiring that there are any unused states in existing control signals.

In particular, output expansion incorporates a logic circuit coupled to two existing signals and an output register. The logic circuit generates a strobe signal in response to a direct transition from one state to another state of the two signals. The host controller is then programmed such that no such direct transition takes place when input/output is being performed to existing peripherals, and such that a direct transition will be detected by the logic circuit when expansion output is being performed. This means that when a transition between the two states is needed to satisfy the existing peripheral, the host controller ensures that the signals sequence through other states and do not go directly between the two states that are detected by the logic circuit.

In the case of input expansion, a logic circuit is coupled to two existing signals and an input buffer. The logic circuit activates an enable signal in response to a direct transition from one state to another state of the two signals, and deactivates the enable signal in response to a transition to a third state of the two signals. As with output expansion, the host controller is programmed such that the first detected direct transition will not take place when input/output is being performed to existing peripherals. Additionally, the host controller is programmed to generate the two transitions detected by the logic circuit when expansion input is being performed.

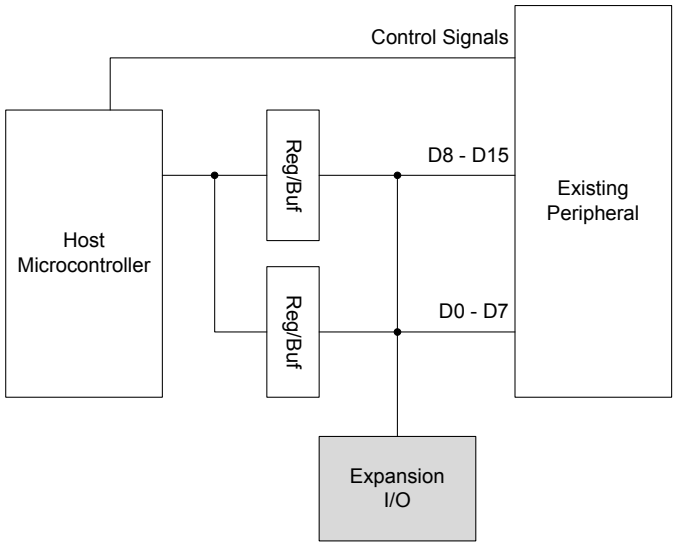

Figure 1

Embedded Environment

Figure 1 illustrates an embedded system in which the expansion circuit can be incorporated. The host microprocessor or microcontroller is responsible for controlling the main functions of the embedded system. The existing peripheral is one or more existing input or output devices. Data signals D0 – D15 are 16 signals that are utilized by the existing peripheral for output of data from the host or input of data to the host. The intervening circuits can be output registers, input buffers, or bidirectional elements incorporating both registers and buffers. The control signals are strobe, enable or select lines that are utilized to communicate between the host and the existing peripheral.

The Expansion I/O circuit in Figure 1 provides additional output signals and is coupled only to data lines of the embedded system and not to any control signals. As will be described in detail below, strobe and enable signals are generated internal to the expansion I/O device based on the sequencing of data on the data lines. The expansion I/O device operates in the embedded system of Figure 1 with no additional wiring and no changes to the existing peripheral. Changes are only needed to the firmware and/or software that controls the sequencing of data from the microcontroller. The only requirement is that the expansion I/O device be coupled to two signals that are continuously driven by the host and that the existing peripheral is not sensitive to the sequencing of data on those signals. It is not necessary that those two signals are not fully utilized by existing peripheral.

Figure 2 illustrates an example of data sequencing for the host microcontroller as it currently exists for performing data output. Signals D0 through D7, and the Strobe or Enable signal are coupled to the host and to the existing peripheral. In order to output data to the existing peripheral, the host microcontroller changes the data on signals D0 through D7 and then generates a transition on the Strobe signal. This can be illustrated as the following sequence:

A: OUT = New_Data<7:0>

B: STROBE

The steps A and B correspond to the dotted lines labeled A and B in Figure 2. Alternatively, to read data from the existing peripheral, the host activates the enable signal, reads the data on signals D0 through D7 and then deactivates the enable signal.

Figure 2 also illustrates the state sequencing of the two least significant signals D0 and D1 in the group D0 through D7. Figure 2 illustrates that since there are no restrictions on the contents of the data previously on signals D0 and D1 with respect to the new state of signals D0 and D1, all state transitions are possible. That is, when the host microcontroller changes the state of signals D0 and D1 from the previous state to the new state at step A, a transition from any of the four possible previous states to the four possible new states can take place.

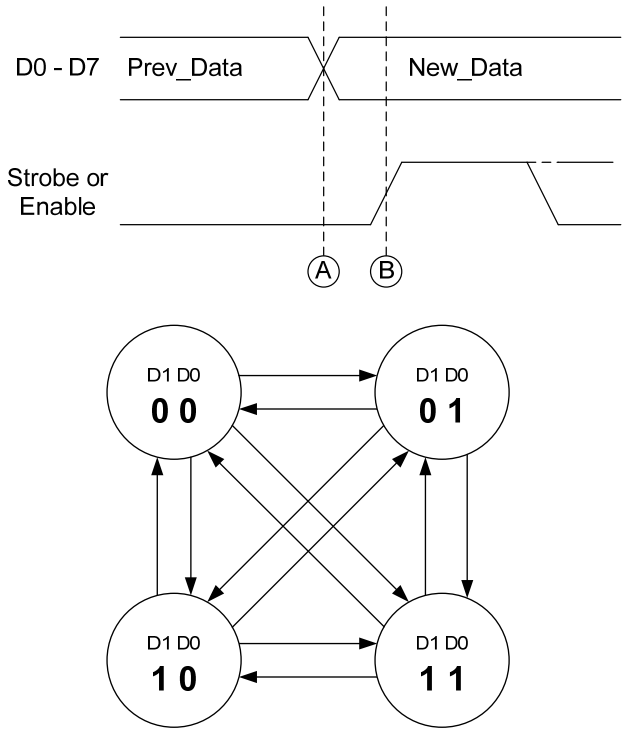

Figure 2

The principle of the expansion circuit described in this paper is that by applying special constraints to the sequencing of existing data output, some of the arcs illustrated in the transition diagram of Figure 2 can be eliminated. The existing peripheral is not affected by the modification to the sequencing because no states are eliminated, only arcs. That is, the existing peripheral waits for the strobe signals before it latches output data, so it does not care if state transitions are not direct and go through other states. Thus, arbitrary data can still be written to existing output devices. However, this elimination of some of the arcs associated with existing data output means that one or more arcs can be reserved for expansion data input and output. This concept will now be illustrated in detail in connection with various examples.

Expansion Input and Output

Referring now to Figure 3, the changes necessary for support of existing data output are illustrated. Figure 3 illustrates a timing diagram for how data is sequenced in connection with a write from the host microcontroller to the existing peripheral. The basic concept behind Figure 3 is that the host

microcontroller is programmed to prevent D1 from changing states when D0 is low. That is, D1 is allowed to change states only when D0 is high. To guarantee this requires a three step sequence that can be described as follows:

A: OUT = Prev_Data<7:1> ‘ 1

B: OUT = New_Data<7:1> ‘ 1

C: OUT = New_Data<7:0>

Note that the single quote character: ' above refers to the concatenation operation. The steps A, B and C correspond to the dotted lines labeled A, B and C respectively in Figure 3. These three steps can be considered a modification of the individual step A from Figure 2. That is, rather than just writing out the new data in one step, as was previously performed and is illustrated in Figure 2, a new three-step sequence is used.

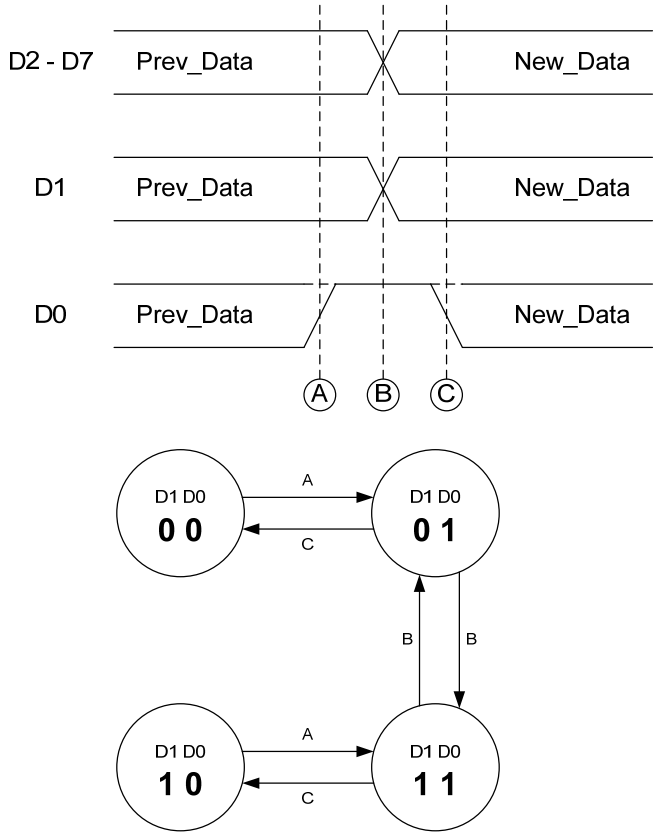

Figure 3

By performing this three-step sequence when data is written to output signals D0 and D1, certain state transition arcs that were present in Figure 2 are no longer present. This is illustrated in Figure 3. The arcs are labeled with A, B and C based on the steps that they correspond to and described above. Note that regardless of which of the four states is the original state and which of the four states is the final state, there are no transitions between the 00 state and the 10 state. Thus all states are accessible and no changes are needed to the existing peripheral for it to operate in the new configuration as illustrated in Figure 3 from the previous operation as illustrated in Figure 2.

While the three-step sequence of Figure 3 is longer than the single step of Figure 2, in many cases the additional time and complexity are negligible. If the time for the additional instructions and cycles is small compared to the frequency with which the data signals are changed, the overhead imposed would be minimal. In some cases, the host microcontroller may need to save the state of the previous data in an internal register so that it can change only D0 without affecting D1. Note that the steps A, B and C described above can consist of explicit actions that are always taken, regardless of the values of the previous and new data, or alternatively, they could be actions that are taken conditional on them being necessary. For example, if the previous state of D0 is a logic 1, then A need not actually be performed. Thus, the host microcontroller could test the previous state of D0 and only take step A if necessary. In some cases this may be preferable than always executing step A even when not necessary.

Rather than having the host microcontroller perform the three-step sequence of Figure 3 as an instruction sequence, dedicated hardware could instead perform this sequencing. In this case, the dedicated hardware would need a way to know if a change in output signals were being made for the purpose of supporting an existing peripheral and would go through the sequence discussed above.

Modifying the existing output from the host microcontroller so that certain state transition arcs are eliminated is only the first half of the solution. The second half is to cause one or more of those eliminated arcs to take place when expansion data is output or input. Figure 4 illustrates one example of such a mechanism. The steps in Figure 4 can be described as follows:

A: OUT = Prev_Data<7:1> ‘ 1

B: OUT = Expansion_Data<7:2> ‘ 01

C: OUT = Expansion_Data<7:2> ‘ 00

D: OUT = Expansion_Data<7:2> ‘ 10

E: OUT = Expansion_Data<7:2> ‘ 11

F: OUT = New_Data<7:1> ‘ 1

G: OUT = New_Data<7:0>

The steps A through G correspond to the dotted lines labeled A through G respectively in Figure 4. The seven-step sequence of Figure 4 guarantees that the step labeled D, which is the state transition from the state 00 to the state 10, takes place in a controlled manner. Figure 4 also illustrates the state transition diagram for the timing diagram with the arcs labeled with the steps to which they correspond.

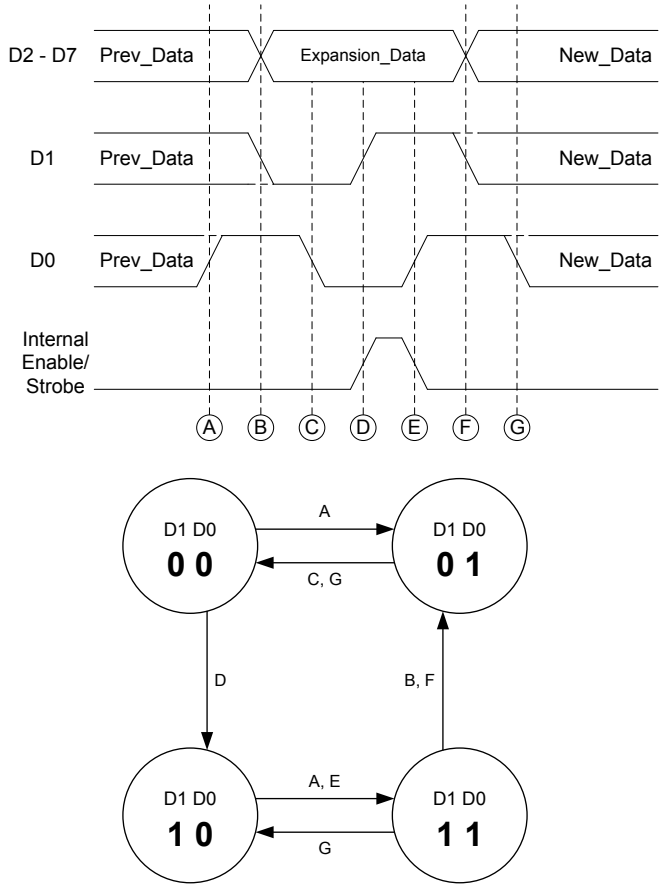

Figure 4

By setting up the expansion output data on signals D2 through D7, and then sequencing the signals D0 and D1 to generate the transition of step D, expansion output of data is accomplished. It is then only necessary for dedicated circuitry present in the expansion I/O device to recognize this transition and generate an internal strobe signal that can be used to latch the data on signals D2 through D7. The use of the high six bits in an eight bit output is only one of many possible examples. Alternatively, more than six bits of output can be accommodated by latching other signals, for example data bits D8 through D15 from the host microcontroller. It would also be possible to utilize two other signals output from the host microcontroller besides D0 and D1. It is only necessary that the data lines utilized are controlled by a single source, so that the transitions on them can be controlled. Note that as describe above with reference to Figure 3, it would be possible to test the previous and new states of D0 and D1 and only take action when necessary. For example, if the new state of D0 is a logic high, then there is nothing to do at step G, so it can be eliminated.

It is also important to note that step E can be used to signal the end of the expansion I/O cycle. This allows the internally generated signal to be used as a buffer output enable for the input of expansion data. Thus, rather than putting data on signals D2 through D7, host microcontroller can float those signals and read the contents of the signals at step E which would be sourced by the expansion I/O device. In alternative implementations, both input and output of data can be accomplished by utilizing an additional signal to indicate the direction of data flow. For example, D2 could be utilized by the expansion I/O device so that if it is low at the time corresponding to step D in Figure 4, an output cycle is generated, while if it is high, an input cycle in initiated.

Note that it would also be possible to utilize the state transition arc going in the opposite direction, from state 10 to state 00, instead of or in addition to the arc going from state 00 to state 10. Since both of these arcs were eliminated in the sequence illustrated in Figure 3, either or both could be used.

Figure 5 illustrates a simplified sequencing of steps that can be used for expansion input and output. The seven steps of Figure 4 can be replaced in certain circumstances with four steps that can be described as follows:

A: OUT = Expansion_Data<7:2> ‘ 00

B: OUT = Expansion_Data<7:2> ‘ 10

C: OUT = Expansion_Data<7:2> ‘ 11

D: OUT = New_Data<7:0>

The steps A through D correspond to the dotted lines labeled A through D respectively in Figure 5. The four-step sequence of Figure 5 guarantees that the step labeled B, which is the state transition from the state 00 to the state 10 takes place in a controlled manner. Figure 5 also illustrates the state transition diagram for the timing diagram with the arcs labeled with the steps to which they correspond.

The reason that the seven steps of Figure 4 can be replaced by the four steps of Figure 5 is that the assumption is made in Figure 5 that signals are sufficiently free from noise that they do not experience bounce as detected by the expansion I/O circuit. That is, the assumption is made that when a signal changes state, going from either high to low or low to high, it does so in a way that allows it to be detected as a single, clean, transition. The validity of this assumption depends on the hysteresis and frequency response of the expansion I/O circuit,

the noise on the data signals as well as other factors. This assumption can often be safely made in well-designed digital systems and significantly simplifies the burden on the host microcontroller.

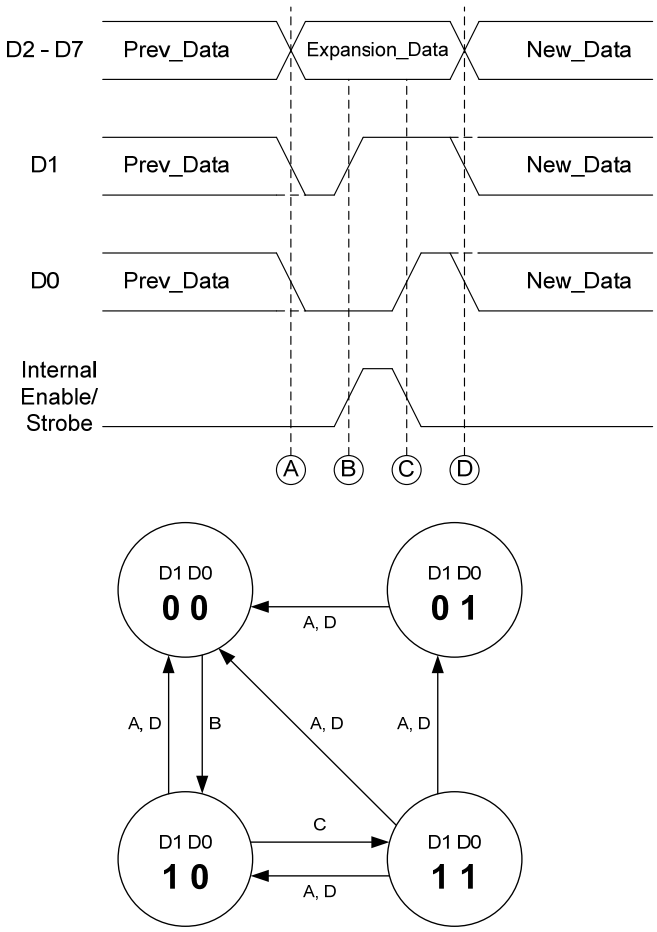

Figure 5

Note that the example of Figure 4 does not make the no-bounce assumption. If each transition of Figure 4 bounces, it will still be the case that the 00 to 10 transitions take place at a single point and in a controlled manner. This is because there is no step in Figure 4 in which both D0 and D1 are changing states at the same time. Figure 5, by contrast, has the property that both D0 and D1 are potentially changing states in steps A and D. However, it is important to note that although the example illustrated in Figure 5 does make the no-bounce assumption, it does not make that assumption that D0 and D1 change states at the same time in steps A and D, as this is generally difficult if not impossible to guarantee. For example, if the previous states of D0 and D1 were 11, step A would change both of these states to 00. The change from 11 to 00 could be detected as a change from 11 to 10 to 00, or from 11 to 01 to 00, or directly from 11 to 00. Each of these possibilities is contemplated by the example of Figure 5. But the no-bounce assumption guarantees that there are no spurious transitions, and thus that the 00 to 10 transition occurs only in step B, when the host microcontroller is ready.

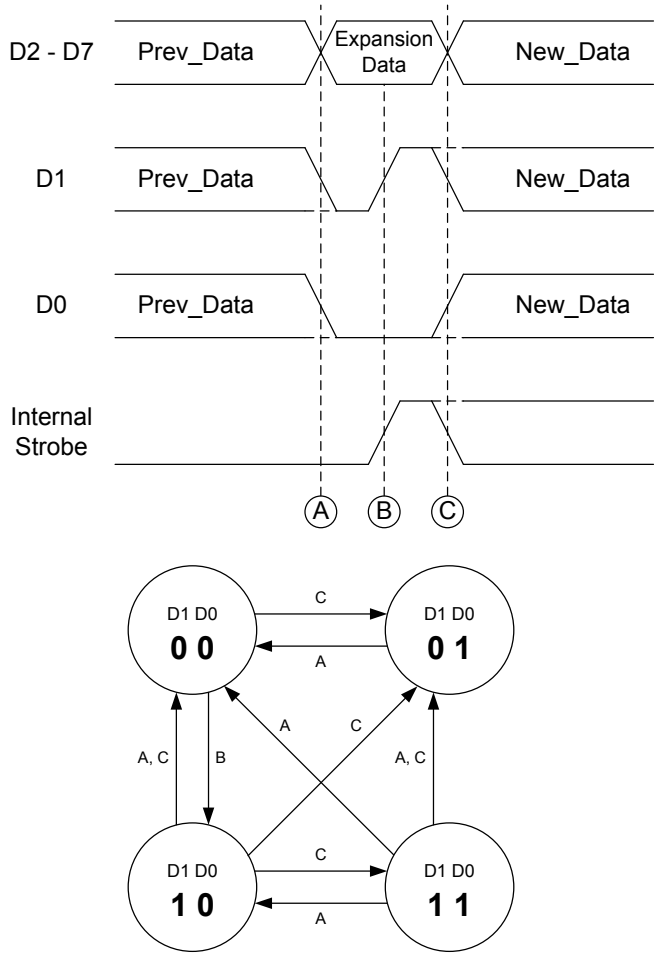

Figure 6

Figure 6 illustrates a further simplified sequencing of steps that can be used for expansion output only. The four steps of Figure 5 can be replaced in certain circumstances with three steps that can be described as follows:

A: OUT = Expansion_Data<7:2> ‘ 00

B: OUT = Expansion_Data<7:2> ‘ 10

C: OUT = New_Data<7:0>

The steps A, B and C correspond to the dotted lines labeled A, B and C respectively in Figure 6. As with Figure 5, the three-step sequence of Figure 6 guarantees that the step labeled B, which is the state transition from the state 00 to the state 10 takes place in a controlled manner. Figure 6 also illustrates the state transition diagram for the timing diagram with the arcs labeled with the steps to which they correspond.

The simplification of Figure 6 is that the step utilized for indicating the end of the expansion I/O sequence has been

eliminated. Thus, if the new states of D0 and D1 are equal to 0 and 1 respectively, the will be no change in state at step C, and the expansion circuit will have no indication that the expansion cycle has ended. This is not a problem for data output, since the expansion circuit is only concerned with generating a strobe, but it will not work for data input, since in that case it is necessary for an enable signal to stop data input at the end of the expansion cycle. Thus, the implementation described by Figure 6 can only be used for expansion data output.

Expansion Circuitry

Figure 7 illustrates an example of circuitry to implement the expansion circuit described in this paper. In this example six bits of data output are provided. The data inputs to the register are coupled to six data lines D2 through D7 from the host microcontroller. The data outputs from the register are coupled to expansion output signals Q0 through Q5. Data lines D0 and D1 from the host microcontroller are coupled to a circuit consisting of a delay element and a three-input AND gate with two inverting inputs and one non-inverting input. The three-input AND gate performs the logical function that can be written as: **/D0 * D1 * /(DELAYED_D1)**. Thus, the output of the AND gate is high when and only when D0 is low, D1 is high and the output of the delay element is low.

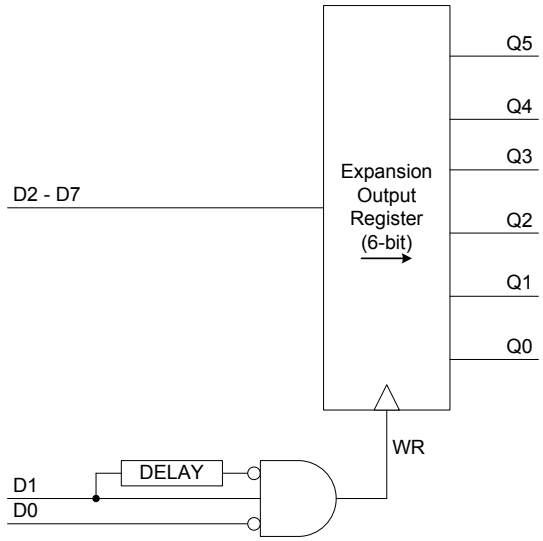

Figure 7

The function of the delay element and the AND gate in Figure 7 is to detect the positive going edge of D1 when D0 is low. This corresponds to the 00 to 10 state transition that is labeled step D in Figure 4 and step B in Figures 5 and 6. When this transition occurs, the output of the AND gate will momentarily go high. The output of the AND gate will only go high at this point and will not go high at any other step of Figures 3, 4, 5 or 6; no other step satisfies the circuit conditions.

The fact that no step in Figure 3 will cause a high output on the AND gate means that the host microcontroller can output data to the existing peripheral without affecting the output register in Figure 7. Thus, arbitrary output to existing peripherals does not affect expansion peripherals.

The delay through the delay element must be chosen long enough to allow the output of the AND gate to fully transition low to high sufficient to cause the output register to be clocked. The length of the low to high to low transitions from the AND gate is approximately the length of time for a signal to propagate through the delay element. The actual time depends on factors such as the minimum and maximum propagation delays, the associated rise and fall times, the input thresholds and the loading on the outputs of the delay element and the AND gate. In practice, the delay element and the AND gate must be designed carefully to guarantee a reliable positive edge on the output of the AND gate in all cases.

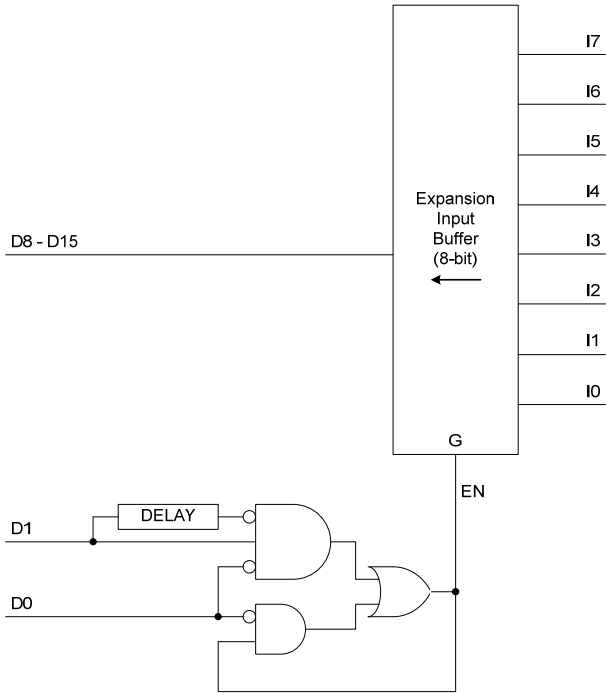

Figure 8

Figure 8 illustrates an example of circuitry to implement input expansion in which eight bits of data input are provided. The data outputs from the input buffer are coupled to eight data lines D8 through D15 to the host microcontroller. The data inputs to the input buffer are coupled to expansion input signals I0 through I7. Data lines D0 and D1 from the host microcontroller are coupled to a circuit consisting of a delay

element, a three-input AND gate, a two input AND gate and an OR gate.

The three-input AND gate performs the logical function that can be written as: **/D0 * D1 * /(DELAYED_D1)**. Thus, the output of the three-input AND gate is high when and only when D0 is low, D1 is high and the output of the delay element is low. The two-input AND gate performs the logical function that can be written as: **/D0 * EN**. Thus, the output of the two-input AND gate is high when D0 is low and EN (the output of the OR gate) is high. The OR gate performs the logical function that is the logical OR of its inputs. That is, the output of the OR gate is high when either of its inputs are high.

The function of the delay element and the three-input AND gate in Figure 8 is the same as the corresponding circuit elements in Figure 7, i.e. to detect the positive going edge of D1 when D0 is low. This corresponds to the 00 to 10 state transition that is labeled step D in Figure 4 and step B in Figures 5 and 6. The additional circuit elements, the two-input AND gate and the OR gate are to latch the state of the output of the three-input AND gate until D0 goes high. This feature causes the output of the OR gate, which is the signal labeled EN in Figure 8, so stay high until the 10 to 11 state transition, which is the labeled step E in Figure 4 and step C in Figure 5.

The delay through the delay element must be chosen long enough to allow the output of the OR gate to transition low to high and to allow the feedback path through the two-input AND gate to latch the signal in the high state. This means that the minimum propagation delay through the delay element must be greater than the sum of the maximum propagation delays through the three-input AND gate, the OR gate and the two-input AND gate. In practice, the delay element and the gates must be designed carefully to guarantee that the 00 to 10 transition is latched reliably in all cases.

Note that the circuit shown in Figure 8 can be used with expansion output as well as expansion input. That is, the circuit driving the EN signal in Figure 8 could replace the circuit driving the clock input in Figure 7. The positive going edge on the output of the OR gate would then be used to clock the output register. In practice, this may present a more robust and reliable design even if the high to low transition on the output of the OR gate is ignored. This is because the circuit of Figure 8 may have more simplified design constraints on the delay element.

Note that even in the case of a Figure 6 data sequencing implementation, which works for data output only, a circuit utilizing the circuitry shown in Figure 8 in conjunction with an output register can be utilized. In that case, the high to low transition on the output of the OR gate may not occur until a subsequent data output, which may be much later in time than the sequence of Figure 6. This will not be important for data output as only the low to high transition is needed to clock the output register.

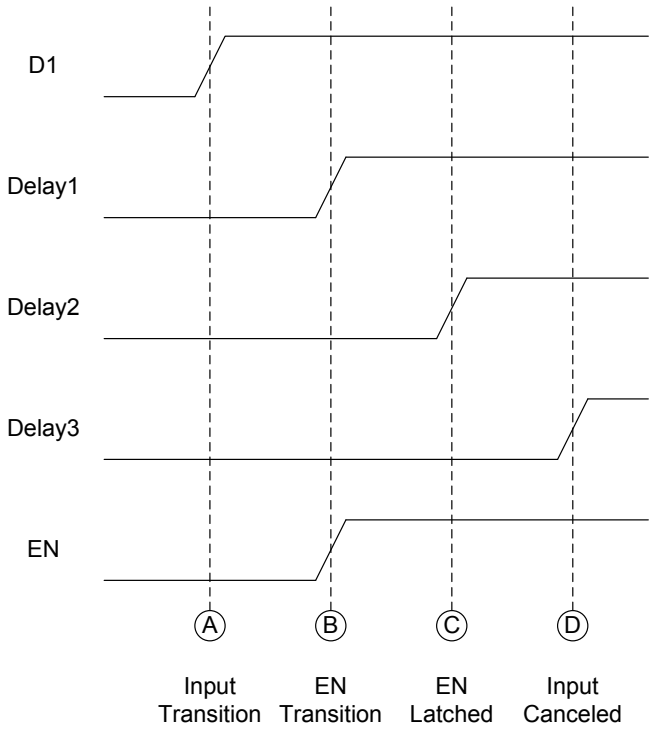

Figure 9

One option for implementing the circuit consisting of the delay element and gates shown in Figure 8 is to utilize a PAL (programmable array logic) device. Figure 9 illustrates a timing analysis of a portion of the circuit shown in Figure 8 when implemented in a PAL device according to the following equations:

/DELAY1 = /D1

/DELAY2 = /DELAY1

/DELAY3 = /DELAY2

/EN = RESET + D0 + (/D1 * /EN) + (DELAY3 * /EN)

These equations implement the equivalent of the logic elements shown in Figure 8. The internal circuitry of a PAL device constitutes a programmable AND array followed by a fixed OR array. If we model the PAL device has having a fixed combinatorial delay from input to output, the timing diagram in Figure 9 can be derived for these equations.

The timing analysis of Figure 9 represents an expanded view of the 00 to 10 state transition that is labeled step D in Figure 4 and step B in Figures 5 and 6. The PAL equations shown above and analyzed in Figure 9 represent an implementation in which the delay element uses three PAL outputs and the gates use a forth PAL output. If we assume that D1 goes through a low to high transition at step A in Figure 9, then by step B, output EN will go through a low to high transition. One propagation delay later, at step C, the output EN is latched due to the feedback of EN. Finally, one propagation delay later, at step D, the output of the third delayed output goes high, which will cancel the effect of the D1 input. This point marks the end of the detection of the low to high transition on D1.

Figure 9, and the PAL equations shown above, illustrate a conservative implementation in which there is a one propagation delay safety margin between when the delay element output goes high and when the output is positively latched. This guarantees reliable operation even in the event of variation in propagation delay time. In an alternative implementation, only two outputs are utilized for the delay element, and the **DELAY2** input is used in the equation of **/EN**. This implementation has the advantage that it utilizes one less PAL output. In most circumstances, this will still result in a reliable and robust design in which the D1 low to high transition is latched, since D1 would be cancelled at approximately the same time that EN is latched. This requires that the propagation delay through the PAL is consistent for different outputs. In an alternative implementation, it may even be possible to utilize a single PAL output for the delay element, although such a design may not reliably latch EN.

Conclusions

In this paper we have discussed a mechanism to expand the input and/or output of an embedded system without relying on any control lines. By utilizing two bits of existing output and by reprogramming the firmware or software to control the transitions on those two bits before being changed, it is possible to build circuitry which can distinguish between output to existing peripherals and expansion input or output. Thus, the circuitry presented in this paper allows a retrofit of an existing system by an expansion circuit that attaches only to the data lines.