APPLICATION NOTE

A Complete Discussion of the S²Cwire™ Single-Wire Interface With tLAT Specification

Introduction
The S²Cwire™ single-wire interface offers the simplest control possible for programmable power I/C devices. Using only a single wire, the S²Cwire interface is designed to be driven by a wide range of systems in a simplified and flexible way. This interface benefits system designers by bringing to market digitally programmable, high performance power I/Cs with minimal pinouts and tiny packaging.

S²Cwire is easily integrated into a wide range of portable systems. The interface does not have a strict timing protocol, simplifying integration for software systems and eliminating any need for specialized hardware. In its entirety, the interface is simple, flexible, and compatible with processor chips found in nearly all of today's portable systems.

This Application Note provides details about programming Skyworks power management I/Cs with the S²Cwire single-wire interface. Insight is also provided for system integration.

S²Cwire Details
S²Cwire is not defined by a strict timing and signaling scheme, but rather by a simple mechanism of toggling a line to program an S²Cwire device. By avoiding strict timing requirements, the S²Cwire interface is kept flexible. The method for programming a device is simple; it consists of toggling the line of an I/O port so that rising edges are received by the EN/SET pin of an S²Cwire device. This can be done with the digital I/O ports found in many processors, or any other digital or analog output capable of sending a line high and low in a pulsed fashion. Toggling the line submits edges to the S²Cwire device, where each rising edge is counted.

Timing is flexible due to the interface’s generous tolerance level. The rising edges that program a part can be fast, slow, or inconsistently spaced. Timing of the edges is bound only by the loose timing specification. The S²Cwire interface is simple to integrate with complex, multi-tasking systems, as well as slow, isolated subsystems dedicated to system management.

Voltage Levels
Several voltage level specifications are associated with the S²Cwire serial interface. The high-impedance input at the EN/SET pin detects logic high and low levels when the high and low voltage level thresholds are achieved for the specified times. The high and low threshold levels are clearly defined in the product Data Sheets as a minimum and a maximum specification, respectively. Table 1 shows an example of the specifications found in most S²Cwire device Data Sheets.

For VIL, the enable low threshold is given as 0.4 V. This means that the applied enable low signal must be 0.4 V or lower. To submit a set of rising edges, the line should be taken below 0.4 V and then above 1.4 V in a repetitive fashion. See Figure 1 for an illustration of this process.

Table 1. Example of EN/SET Threshold Specifications

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Conditions</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIL</td>
<td>Enable threshold low</td>
<td>VIN = 2.7 V</td>
<td></td>
<td>0.4</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>VIH</td>
<td>Enable threshold high</td>
<td>VIN = 5.5 V</td>
<td>1.4</td>
<td></td>
<td></td>
<td>V</td>
</tr>
</tbody>
</table>
Figure 1. Voltage Level Thresholds

Table 2. Example of EN/SET Timing Specifications

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Conditions</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>tEN_SET_LO</td>
<td>EN/SET low time</td>
<td>$V_{IN} = 2.7 \text{ V}$</td>
<td>0.3</td>
<td></td>
<td>75</td>
<td>$\mu\text{s}$</td>
</tr>
<tr>
<td>tEN_SET_HI_MIN</td>
<td>Minimum EN/SET high time</td>
<td>$V_{IN} = 5.5 \text{ V}$</td>
<td>50</td>
<td></td>
<td>75</td>
<td>ns</td>
</tr>
<tr>
<td>tEN_SET_HI_MAX</td>
<td>Maximum EN/SET high time</td>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>$\mu\text{s}$</td>
</tr>
<tr>
<td>tOFF</td>
<td>EN/SET off timeout</td>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>$\mu\text{s}$</td>
</tr>
<tr>
<td>tLAT</td>
<td>EN/SET latch timeout</td>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>$\mu\text{s}$</td>
</tr>
</tbody>
</table>

Timing

A few timing specifications are associated with the S‘CWIRE interface. These relate to how fast or slow the interface can detect rising edges, as well as when to expect the data registers to update. Internally, a counter counts the rising edges and, after the line has been held high, the data is latched and the output is updated.

Table 2 provides an example of the EN/SET timing specifications found in most S‘CWIRE product Data Sheets. The specifications that correspond to edge detection are the tEN_SET_LO and tEN_SET_HI. For determining the tLO time, the Data Sheet gives a minimum and a maximum specification for tEN_SET_LO. This results in a straightforward set of limits. To ensure edge detection, be certain that the tLO time is at least the minimum time and no greater than the maximum time. If edges are submitted outside of this range, there is risk that the edges are not detected or that a latch timeout occurs when not intended (see Figure 2).

For determining the tHI time, the Data Sheet gives two specifications. There is a typical value for tEN_SET_HI_MIN and a maximum value for tEN_SET_HI_MAX. To determine the minimum tHI time, double the typical value for tEN_SET_HI_MIN. To determine the maximum value for the tHI time, use the maximum value from the tEN_SET_HI_MAX specification. If edges are submitted outside of this range, there is risk that the edges are not detected or that a latch timeout occurs when not intended.

The Data Sheet gives two additional specifications that detail the S‘CWIRE interface. One is tLAT, which is given as a maximum specification. This means that after submitting data and then holding the line high, the latch timeout is guaranteed to occur by this time. Typically, the tLAT timeout occurs more towards the middle of the window created by tEN_SET_HI_MAX and maximum tLAT (e.g., between 75 $\mu\text{s}$ and 500 $\mu\text{s}$). The output updates after tLAT, as illustrated in Figure 2.

The final relevant timing specification is tOFF. The tOFF time is given as a maximum specification. This means that after taking the line low, the part is guaranteed to enter shutdown within this time. Note that when the part is shut down, the data register is reset to zero, as illustrated in Figure 3.
Programming

Programming an S²Cwire device is simple. This section details what is involved and is intended to be general so that it applies to all S²Cwire devices. A basic block diagram for a sample device, used to facilitate the programming discussion, is shown in Figure 4. The available operating modes for this device are presented in the same fashion as found in the Data Sheet for a real device. This example operates as a constant voltage source where the output voltage is programmable through the S²Cwire interface. There are 16 programmable settings available for this device, as defined in Table 3.

In the Data Sheet of an S²Cwire device, the complete functionality of the device is listed in a Table similar to that shown in Table 3. Programming an output voltage is accomplished by submitting a burst of edges corresponding to the desired setting and then holding the EN/SET line high. The device remains in this programmed state for as long as the EN/SET signal remains high or until it is reprogrammed. Taking EN/SET low shuts down the device and resets the data register. If no edges are clocked in, but EN/SET is taken high, the device operates in the default state (Data 1).

Following is a discussion of a few simple programming examples to illustrate some of the details of operation. As a first example, suppose an event occurs and it is necessary to enable the constant voltage output to 4.25 V. From Table 3, 4.25 V corresponds to Data 4. Therefore, four edges are required to turn on the output to this setting. To bring the device out of shutdown and turn on the voltage output to 4.25 V, clock-in four edges for Data 4 and hold the line high. After the latch timeout, the output updates to the programmed output level, as illustrated in Figure 5.

Table 3. Mode Settings

<table>
<thead>
<tr>
<th>Data</th>
<th>Voltage (V)</th>
<th>Data</th>
<th>Voltage (V)</th>
<th>Data</th>
<th>Voltage (V)</th>
<th>Data</th>
<th>Voltage (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.00</td>
<td>5</td>
<td>4.00</td>
<td>9</td>
<td>3.00</td>
<td>13</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>4.75</td>
<td>6</td>
<td>3.75</td>
<td>10</td>
<td>2.75</td>
<td>14</td>
<td>1.75</td>
</tr>
<tr>
<td>3</td>
<td>4.50</td>
<td>7</td>
<td>3.50</td>
<td>11</td>
<td>2.50</td>
<td>15</td>
<td>1.50</td>
</tr>
<tr>
<td>4</td>
<td>4.25</td>
<td>8</td>
<td>3.25</td>
<td>12</td>
<td>2.25</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>
Suppose the device is operating with the output set to 4.25 V and an event occurs signaling that it needs to change to 5 V. This is accomplished by immediately clocking-in the appropriate number of edges (i.e., EN/SET does not need to be pulled low to reset the data register first). To change the output voltage from 4.25 V to 5 V, only 1 edge is needed because 5 V corresponds to data setting 1, as illustrated in Figure 6.

Lastly, the device needs to be put into shutdown. The device has been in an operating state because the EN/SET pin has been held high. Simply bring the EN/SET line low to enter shutdown mode. After taking the line low, there is a delay time \( t_{OFF} \) before shutdown mode is entered, as shown in Figure 7. Note that shutdown resets the data register to zero.

A more general timing diagram, as shown in Figure 8, appears in many Data Sheets. All of the information that has just been covered is captured in this diagram. The general idea is that the user submits a set of edges to program an S2Cwire device. After the delay time \( t_{OFF} \), the device is programmed and is updated to a particular operating state. To put the part into shutdown mode, simply take the EN/SET pin low. When \( t_{OFF} \) has expired, the part enters shutdown and the data register is reset.

In summary, programming or re-programming an S2Cwire device simply requires clocking in edges for the desired setting and then holding the EN/SET line high. To ensure that S2Cwire programming is successful, be certain that the signals fall within the timing specifications and that the signals achieve the voltage level thresholds. To turn the part off, the EN/SET pin is simply held low.
Software Integration

There are a number of practical ways to integrate S²Cwire functionality into embedded software systems. Many of today’s cell phones and handheld devices are built around low-power, high performance 16- and 32-bit processors that are running a Real-Time Operating System (RTOS) and application Operating System (OS). These systems are balancing heavy loads with many hard, real-time deadlines that cannot be missed. As a result, careful consideration is given to every additional piece of code that is added to a system. In line with today’s methodologies, S²Cwire is flexible so that it is easy to integrate.

Due to the simplicity of the S²Cwire interface, very little code is needed (see the example in Figure 9). Routines can just write 1s and 0s to an I/O port to toggle the line that is connected to the EN/SET pin of the part. The code can be fast and small, allowing it to be implemented as a high priority task. A high priority task, dedicated to writing sets of edges, can run to completion extremely quickly since the S²Cwire timing specification allows for fast timing. This fast execution time results in minimal impact to the software system.

During the actual burst of edges, the code should execute with high priority. If using a preemptive, multi-tasking system such as an RTOS, a high priority task could be used during the data burst. Then, the wait for tLAT can occur at the application level. Since the S²Cwire interface accepts fast timing, the high-priority task could toggle the I/O line quickly so that CPU time is minimized, as illustrated in Figure 10.

Since the actual data burst is of primary concern, it is useful to estimate the execution time. The calculation focuses purely on the

S²Cwire pulse timing and does not account for any OS latencies. To minimize execution time, the fastest EN/SET timing should be used (300 ns for tLO and 100 ns for tHI). With this timing, the longest execution time occurs for the highest data setting. The highest data setting requires 16 edges. This is the largest burst of edges and should be used to estimate the time. It can be calculated as follows:

\[ t_{RUN} = \text{max\_edges} \times (t_{LO}^{\text{MIN}} + t_{HI}^{\text{MIN}}) \]

\[ t_{RUN} = 16 \times (300 \text{ ns} + 100 \text{ ns}) = 6.4 \mu s \]

From this calculation, it can be seen that the necessary execution time for the longest burst of edges is 6.4 μs.

A slow rate of edges can also be detected by the S²Cwire interface. This gives a task dedicated to S²Cwire a high degree of tolerance so that it can be preempted by more important code. However, a major concern to consider is whether the task has a high enough priority. Using the previous example, the task should have a high enough priority so that control returns within 75 μs (because this is the tEN/SET_HI_MAX and tLO maximum specifications). If the task is kept waiting for more than 75 μs, it could cause a delay within the burst of edges to exceed 75 μs. If tHI or tLO are allowed to exceed 75 μs, a tLAT or tOFF timeout may occur and interfere with the intended programming. For a simplified example, see Figure 11.

The S²Cwire serial interface fits within the general multi-tasking RTOS environments commonly used, as well as the custom-rolled software solutions tailored to a specific application. With little effort, S²Cwire control can be easily integrated into today’s software/hardware platforms.
Summary
The S²Cwire single-wire interface and possible approaches to software integration have been described in this Application Note. The overall discussion is intended to give additional insight into using the interface in the target application and has attempted to clarify the conceptual aspects of the S²Cwire single-wire interface.

With this knowledge, designers can more effectively take advantage of the features and benefits offered by Skyworks digitally programmable, high performance I/C devices.