Course Introduction

Purpose:
• The intent of this course is to provide embedded control engineers with valuable implementation instructions on HCS08 port pins and the Keyboard Interrupt (KBI) module.

Objectives:
• List the key features of the S08 port structures.
• Describe the operation of the keyboard interrupt module.
• Describe techniques used to reduce current draw, reduce electromagnetic radiation, and increase ADC accuracy.

Content:
• 21 pages
• 4 questions

Learning Time:
• 35 minutes

This course addresses the parallel I/O and Keyboard Interrupt Module (KBI) that is on many of the microcontrollers in the S08 family. The intent of this course is to provide embedded control engineers with valuable implementation instructions on HCS08 port pins and the KBI module. It's important to understand the features and some of the common questions that people ask when using an S08. This training course gives you some tricks of the trade for using S08 microcontrollers.

In this course, you will examine the key features of the port structures, including pull-up and pull-down devices, slew rate control, and drive strength control. You will also look at the operation of the keyboard module, including features, registers, and Stop mode behavior. You will learn various techniques used to reduce current draw, reduce electromagnetic radiation, and to increase ADC accuracy. By understanding appropriate pin selection and configuration, you can keep ADC error to a minimum.
Let’s begin with a look at a simplified diagram of the pad structure, starting with the three vertical legs. The leftmost leg contains protection diodes. If the pad were ever to be pulled above $V_{DD}$ or below $V_{SS}$, the current would flow through those diodes effectively clamping the voltage to a diode drop above $V_{DD}$ and protecting voltage sensitive circuits of the device. Sometimes, port pins are used to help measure input signals that temporarily go above the rail, such as an AC line or an automobile. This may be the case when trying to monitor the battery voltage, which can vary from 6V when cranking to 24V when charging.

The second leg has pull-up (PU) and pull-down (PD) resistors, which are software-selectable and can be configured on the fly. This enables you to select either pull-ups or pull-downs, but not both at the same time. Instead of having to put external resistors on the printed circuit board, you can make use of the internal resistors to reduce the cost of the end application. Eliminating components also increases reliability.

The third leg contains a high drive and a low drive, which appear here as a single-transistor entity. In reality, they are multiple transistors that can be switched on or off, depending upon the drive strength and the slew rate control. Current-limiting resistors are also associated with the third leg. While they are small in size, they offer enough series resistance to protect the device from temporary short-circuits.

Software can individually configure slew rate control for the output buffers. If a pin is configured as an output, and you don’t need a fast-rising or falling edge, you can turn on the slew rate control, which can greatly help with the electro-magnetic emissions.

Configurable drive strength is also available. High drive strength can be useful for, driving an LED to keep a certain intensity. Low drive strength can be useful if you’re only driving a digital load such as another IC or a sensor. Note that regardless of whether the pin is being controlled by the port registers or by another module, pin controls (such as the pull-ups, the slew rate control, and the drive strength control) will be active. For high drive strength, the typical source (or sink) current is $\geq 10$ mA at 500 mV off rail.
## Configurable Input/Output Pins

- **Registers Associated With I/O**

<table>
<thead>
<tr>
<th>Parallel I/O Reg.</th>
<th>Mapped to Direct Page</th>
<th>Mapped to High Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>PTxD</td>
<td>R/W value of pin</td>
</tr>
<tr>
<td>Data Direction</td>
<td>PTxDD</td>
<td>Set pin as input or output</td>
</tr>
</tbody>
</table>

### Pin Control Reg.

<table>
<thead>
<tr>
<th></th>
<th>Mapped to Direct Page</th>
<th>Mapped to High Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pull-up Enable</td>
<td>PTxPE</td>
<td>Enables pull-up</td>
</tr>
<tr>
<td>Slew Enable</td>
<td>PTxSE</td>
<td>Enables slew rate</td>
</tr>
<tr>
<td>Drive Strength</td>
<td>PTxDS</td>
<td>Set drive strength</td>
</tr>
</tbody>
</table>

**Pin control functions remain enabled – even when other modules govern the pin activity!**

Five registers are associated with each parallel port. These registers are broken into two groups. The first group, called "Parallel I/O", includes the Data and the Data Direction registers. The Data register is used to read the value of an input or writes the value if it is set up as an output. The Data Direction register sets the pins as an input or an output. These two registers are often accessed frequently and are placed in the direct page memory of the S08. This enables them to make use of the more efficient addressing modes associated with the direct page.

The second group of registers comprises the pin control registers. Whether the pin is being used as a parallel I/O or is being controlled by another module, the pin control registers govern the behavior of the pin itself. The pin control registers include: a Pull-up Enable register with a single bit for each of the 8 bits associated with that port; a Slew Enable register that also has a single bit for each of the 8 associated bits, and the Drive Strength register that sets either a low-drive or a high-drive for each of the pins.

Out of reset, the Pull-up, the Slew, and the Drive Strength control registers are all set to their default values. This means that the pull-ups are disabled, the slew rate is disabled, and the drive strength is set at low drive.
The diagram shown here explains the multiplexed functions of many of the I/O pins. On the low pin count devices, to make many functions available, each of the pins must be multiplexed with potential functionality. For example, in the box in the lower-right, a pin is called out. The call letters are PTA0, or Port A0; KBIP0, or Keyboard Interrupt Pin 0; TPMCH0, or Timer Channel 0; ADP0, or A-to-D Pin 0; and ACMP+, or Analog Comparator Plus input signal.

Five different functions are associated with this particular pin. Some circuitry is also associated with controlling the specific function that gains control of the pin if more than one is enabled. The right-most label in the pin name gets the highest priority. In this particular case, the Analog Comparator Plus input has priority if any of the other functions are enabled. Normally, under software control only one function is enabled at a time, so priority may be less important. However, Freescale must set priority to protect the device and to call out the actual operation if two or more functions are enabled at one time.

Let’s look a bit more closely at priorities. If a serial module, such as an SPI or an SCI, is enabled, it will dictate the function of certain port pins. Individual signals that feed into the pad interface circuit control the port pin itself. The mode setting of the serial module would determine the associated function. For example, an SCI can be put into a wired OR configuration. In that particular case, only the transmit pin is used; the receive pin would not be used. If the SCI were enabled in the normal SCI mode, one pin would be set up for transmit, and the other would be set up for receive. The transmit is an output and the receive is an input.

The analog modules are a bit different in that each one has individual control bits associated with enabling a certain channel with that module. With digital modules, the port can be read if it’s an input, and that reading is valid. However, if the pin is configured for an analog function and you try to read that port, the value you get will not correspond to the logic level of the input pin. This is because the digital connections to that node must be turned off to provide a clean analog path.

For timer (or TPM) channels, a function is associated with each pin. Depending on whether a pin is assigned to be an input capture, an output compare, or a PWM, the Pad interface logic will set the state of the port pin as an input or an output.
Some pins have a dedicated input or output function and therefore are not I/O pins; however, there are few of these pins.

The most common dedicated output pin is the Background Debug Mode/Mode Select (BKGD/MS) pin. This pin configures the device out of reset to go either into Background Debug mode or into User mode. Once it is in Background Debug mode, the pin is used for the background communication channel. In this mode, the pin actually performs an I/O function under control of the Background Control (BDC) module. However, the user can only set it up as an output because having the external circuitry driving it high or low might corrupt the moding of the device out of reset.

A few pins are dedicated inputs. On some of the Flash devices, a higher voltage (that is, a voltage greater than V_{DD}) must be supplied to the Flash array for testing. Most S08 MCUs have one pin dedicated for this function. That high-voltage circuitry does not allow the same protection diode to V_{DD}. The alternate protection does not allow output drivers to be sourced from V_{DD}. As a result, some pins are limited as input only when their alternate function is a high-voltage test circuit.
Question

Indicate whether the following statement true or false. Click Done when you are finished.

“Priority determines the specific function that gains control of a pin. The left-most label in the pin name always has highest priority.”

True

False

Consider this question concerning multiplexed functions.

Correct.

The way that priority works is the right-most label in the pin name gets the highest priority if any of the other functions are enabled. Click the forward arrow to continue on to the next page.
Now, let’s look at some of the electrical details of the parallel I/O, beginning with the slew rate control. With slew rate control disabled, into a 50 pF load, we will have a rise and a fall time of typically 3 ns. When the slew rate control is enabled, the transition from a low to a high or a high to a low will be 30 ns.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port rise and fall time (load = 50 pF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slew rate control disabled (PTxSE = 0)</td>
<td>$T_{\text{rise}}$</td>
<td>--</td>
<td>3</td>
<td>--</td>
<td>ns</td>
</tr>
<tr>
<td>Slew rate control enabled (PTxSE = 1)</td>
<td>$T_{\text{fall}}$</td>
<td>--</td>
<td>30</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>
Here is a slew rate control example. There are two pins, Port A0 and Port A1, that transition at exactly the same time. One of the pins has the slew rate disabled; the other one has the slew rate enabled. The pin at the top has no slew rate. In this particular case, the pin is being driven into a 1 kOhm load with the scope as the only capacitance.

You will notice on the top, we have a fairly fast rise time with a small overshoot, and the signal at the bottom doesn't have an overshoot at all. It also has a fairly controlled slew rate. The slew rate can really reduce radiated emissions. Excessive overshoot and subsequent ringing may indicate that insufficient decoupling capacitors have been provided or that the signal trace on the PCB is too long.

There are two things to focus on here: the steepness of the edge as a factor in the radiation and the overshoot. If the overshoot is greater than a diode drop, you can turn on the protection circuit inside the device and a fairly large amount of current can flow. When that diode turns on and the current flows, a lot of radiated emissions can be produced. Therefore, look at all the pins in your design and determine which ones are capable of handling a slow rise time. If a fast edge is not required electrically, you should enable the slew rate control to keep your radiated emissions to a minimum.
Now, let’s examine drive strength performance, looking first at the electrical parameters associated with the pins in the high and the low drive states. At the top, you can see the voltage out low (V_{OL}) for the different drive strengths and the different current values.

Consider the 10 mA example where V_{DD} is greater than 2.7V. The spec guarantees that it will be driven between 0V and 0.5V. Similarly, the V_{OH}, which is V output high, will be within half a volt of V_{DD}. For a 3- or 5-volt device, S08s will have slightly different performance characteristics. It is always best to consult the actual electrical characteristics of the device that you are using.

Another drive strength characteristic to consider is a maximum sink-and-source current for each device. This is dependent upon the size of the die, the number of power pins, and various other things, such that it can vary from device to device. The purpose of the sink-and-source current is to control metal migration of power traces inside the device. Over time, driving too much current can result in small pieces of metal bridging between V_{DD} and V_{SS}. On this particular device, there is a maximum total I_{OL} for all port pins of 60 mA. The user is requested to keep it under 60 mA for any extended duration.

Note that all of the data books show typical drive curves—V\text{out} versus I\text{out}. If you need to drive 20 mA on any one pin, consult the data book to see the expected V\text{out} for that operating point.
Now, let’s look at scope captures that show the impact of drive strength and slew rate. Here you can see the impact of drive strength alone.

Let’s examine the test condition first, and then we’ll look at some of the details of the waveform. This is port A0 and port A1, both with a 330 Ohm resistor tied to $V_{DD} = 3.0V$. When the pin is driving low, it is pulling about 10 mA of current. When the pin is driving high, no load is present. As one would expect, the high drive strength pin in light red is able to pull the load closer to $V_{ss}$. However, it also has a larger overshoot when it goes to the unloaded state.
Now, let's take a look at the impact of both the drive strength and the slew rate. The time per division was changed here, nonetheless, you can still see that the normal drive with no slew rate has a little bit of overshoot. Looking at the high drive strength and slew rate control, you can see that it still has plenty of strength to pull the pin low with a load of 10 mA, and there is also no overshoot. This waveform has a little bit of distortion, but that comes from the digitizing from the oscilloscope and not so much from the actual output of the pad itself.
Parallel I/O Electricals

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal pullup resistors</td>
<td>$R_{PU}$</td>
<td>17.5</td>
<td>--</td>
<td>52.5</td>
<td>kohm</td>
</tr>
<tr>
<td>Internal pulldown resistors</td>
<td>$R_{PD}$</td>
<td>17.5</td>
<td>--</td>
<td>52.5</td>
<td>kohm</td>
</tr>
</tbody>
</table>

### Input Thresholds and Hysteresis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input High voltage ($V_{DD}&gt;2.3V$) (all digital inputs)</td>
<td>$V_H$</td>
<td>$0.70 \times V_{DD}$</td>
<td>--</td>
<td>--</td>
<td>V</td>
</tr>
<tr>
<td>Input High voltage ($1.8 \text{ V} \leq V_{DD} \leq 2.3\text{ V}$) (all digital inputs)</td>
<td>$V_H$</td>
<td>$0.85 \times V_{DD}$</td>
<td>--</td>
<td>--</td>
<td>V</td>
</tr>
<tr>
<td>Input Low voltage ($V_{DD}&gt;2.3V$) (all digital inputs)</td>
<td>$V_L$</td>
<td>--</td>
<td>--</td>
<td>$0.35 \times V_{DD}$</td>
<td>V</td>
</tr>
<tr>
<td>Input Low voltage ($1.8 \text{ V} \leq V_{DD} \leq 2.3\text{ V}$) (all digital inputs)</td>
<td>$V_L$</td>
<td>--</td>
<td>--</td>
<td>$0.30 \times V_{DD}$</td>
<td>V</td>
</tr>
<tr>
<td>Input hysteresis (all digital inputs)</td>
<td>$V_{hys}$</td>
<td>$0.06 \times V_{DD}$</td>
<td>--</td>
<td>--</td>
<td>V</td>
</tr>
</tbody>
</table>

Next, let’s look at some key parallel I/O electrical characteristics: the values of the pull-up and pull-down resistors, and the values of the input thresholds and hysteresis.

The range of the pull-up and pull-down resistors is between 17.5 and 52.5 kilo-Ohms. This range works well for devices such as keypads and buttons, or just pulling the device up without any expected activity on the pin. Most data books provide a typical curve showing the resistor value over temperature and voltage.

Input threshold and hysteresis is a common product support subject. Clients require an understanding of input thresholds in order to configure their systems to handle ground bounce, ground offset, or $V_{DD}$ and $V_{SS}$ differences from one portion of the board to another, or, in some cases, from one board to another.

The table shows ranges for a 3-volt part. Note that there are two ranges: one is above 2.3V, the other is between 1.8V and 2.3V.

For most devices, the switch point will be at half of $V_{DD}$; however, some hysteresis is added to prevent switching noise near the threshold. The added hysteresis is guaranteed to be at least 6 percent, or $.06$ times $V_{DD}$.

For an input, anything above 70% of $V_{DD}$ is guaranteed to be interpreted as a logic 1; anything below 30% of $V_{DD}$ is guaranteed to be a logic zero.
Handling Voltages Above $V_{DD}$

The MCU is capable of handling a small amount of current injection when an input drives above $V_{DD}$ or below $V_{SS}$.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dc injection current $V_{IN} &lt; V_{SS}, V_{IN} &gt; V_{DD}$</td>
<td>$I_{IC}$</td>
<td>-0.2</td>
<td>-5</td>
<td>0.2</td>
<td>mA</td>
</tr>
<tr>
<td>Single pin limit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total MCU limit, includes sum of all stressed pins</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

**TIP:** Separate pins subject to current injection from analog inputs.

Signals being monitored by the MCU sometimes exceed $V_{DD}$. For example, some automotive applications connect switches to the automobile’s battery voltage rather than to $V_{DD}$ of the board. Some industrial applications monitor a building’s AC line voltage. It can be a challenge at times to feed 120V—or 12V—into a 3V or 5V microprocessor. Voltage can also drop below $V_{SS}$, such as when pulling something below ground.

By using a simple resistor divider and understanding the capability of the MCU, you can make S08 devices handle voltages above $V_{DD}$ or below $V_{SS}$.

By choosing the right values for the current-limiting resistor and setting up a resistor-divider network, you can monitor the voltage of either battery or monitor AC line voltages. Just make sure that the continuous voltage into any single pin is under 200 µA and that the total for the entire MCU is under 5 mA. The same applies if you were pulling something below ground, or below $V_{SS}$. 


**Question**

What is the pull-up and pull-down resistance range? Select the response that applies and click Done.

a. 10 kΩ – 100 kΩ  
b. 5 mΩ – 10 mΩ  
c. 500 k – 1 MΩ  
d. 17.5 kΩ – 52.5 kΩ

Done

While it’s still fresh in your mind, let’s review pull-up and pull-down resistance.

Correct.

The range of the pull-up and pull-down resistors is between 17.5 and 52.5 kilo-Ohms. Silicon processing does not enable tight control of resistance in this range.
Warning on Unbonded Pins

Stop \( I_{DD} \) is dependent upon state of I/O.

**NOTE**
Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user reset initialization routine in the application program must either enable on-chip pull-up devices or change the direction of unconnected pins to outputs.

Make certain all unused and unbonded pads have terminations!

Next, we will look at an important issue, one that results in many questions to product support: what do you do with unused pins (that is, pins that are either not used on the printed circuit board or that are not available in the particular package being used)?

Unused pins must be terminated properly to minimize the Stop mode current and the coupling of radiated emissions from other devices. Note that a CMOS input has a very high input impedance and can float either high or low if left unterminated. If the input sits near the transition point and causes internal switching, higher than normal stop \( I_{DD} \) can occur.

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user reset initialization routine in the application program must either enable the pull-up devices or change the direction of the unconnected pins to an output so the pins do not float.

It is very important either to set up any unused pins as an output or to enable their pull-up device. Ensure that unused and unbonded pads have terminations.
Initialization and Pin Selection

- **Software**
  - Immediately configure pins after reset.

- **Hardware**
  - Place high drive pins as close to $V_{DD}/V_{SS}$ as possible.
  - Separate high-activity pins and analog pins as much as possible.
  - Separate any pins potentially exceeding $V_{DD}$ or $V_{SS}$ from any critical and/or sensitive analog inputs.

Here are a couple of hints that can help you optimize your design.

Configure all the I/O pins immediately after reset, even if they will be enabled later for a timer or an SCI module. This prevents drawing excess current or accidentally turning on or off any external device that is controlled by the MCU.

Design any high-drive pins to be placed as close as possible to the $V_{DD}$ and $V_{SS}$ pins. This minimizes the internal Current X Resistance (IR) drop and the loop area. Locate any pins that will be used to drive heavy loads (such as LEDs) or relays as close to $V_{DD}$ as possible.

The second hint relates to high-activity I/O pins, such as SPI clock lines, high-speed timer inputs or outputs, and high frequency clock lines. Separating these high-activity pins as much as possible from any analog pins, or functions, will reduce the error caused by noise coupling into those analog pins. Most S08 devices simplify this choice by allowing analog inputs on many of their pins.

Finally, separate any pins potentially exceeding $V_{DD}$ or $V_{SS}$ from the critical and/or sensitive analog pins. When the voltage exceeds $V_{DD}$, the input protection diodes turn on and current flows through that particular pad. If the current is at a high level, it will let off a field and impact circuits in its general vicinity. The pads provide some built-in isolation, but it is not always enough. You can increase this isolation by moving sensitive I/O pins away from any pin that might be clocked above $V_{DD}$ or below $V_{SS}$.
Keyboard Interrupt Module (KBI)

- **Purpose:** Provide flexible way to interrupt the MCU for external events.
- **Features:**
  - Configurable interrupt source
    - Single interrupt vector
  - Rising or Falling edge selectable
  - Wake-up mechanism from low-power Stop & Wait modes
  - Pin-by-pin enable

Now let’s look at the keyboard interrupt module, shown here in the block diagram.

The keyboard interrupt is a convenient and flexible way of handling and monitoring input activity over eight pins.

Each pin has a separate enable and it is not required that all eight pins of the keyboard module be enabled. A single interrupt vector is used for all eight pins on the device.

Each of the eight pins can be configured for a rising or falling edge, or event capture. The keyboard interrupt module provides a nice wake-up mechanism from Stop and Wait modes. Pin-by-pin enable is available on the keyboard interrupt module.
Let's take a moment to look at asynchronous and synchronous inputs, or, the difference between when the device is running and when it's not.

When the CPU is running, a synchronous system exists in which a clock dictates all activity and inputs are sampled at a certain time relative to the bus clock.

When the part goes into Stop mode, all of the clocks are shut off and the part has no clock to sample and align input signals. In Stop mode, input signals occur asynchronously and the RESET, the IRQ and KBI pins are triggered asynchronously.

A separate line item for keyboard interrupt pulse widths will appear in each data book for the asynchronous path. On this device, that value is 100 ns. Any pulse that comes in that is wider than 100 ns will be detected.

However, when the CPU is running, a sampling technique is used to capture the state of input pins. In order to guarantee the input signal is captured, a pulse width must be 1.5 t-cycles, where t-cycle is the time of one clock period for the device. For example, at 10 MHz, t-cycle would be 100 ns. Or, on this device, the synchronous path would be 150 ns wide.
Question

What valuable function does the KBI perform? Select all responses that apply and click Done.

a. Software filtering of input signals
b. Supports the coprocessor
c. Monitors input activity across the pins
d. Provides a wake-up mechanism from Stop and Wait modes

Done

Take a moment to review the keyboard interrupt module.

Correct.

The KBI handles and monitors inputs over 8 pins and acts as a wake-up mechanism from low-power Stop and Wait modes.
Next, let’s look at the three registers associated with the KBI module: the Status and Control register, the Pin Enable register, and the Edge Control register.

The Status and Control register has four bits. A separate Interrupt Flag bit is set when an interrupt event occurs. Writing to the Interrupt Acknowledge bit clears the flag. The Interrupt Enable bit enables the entire module to have interrupts.

The Status and Control Register’s Mode bit configures all of the eight pins associated with the module in one of two modes: either Edge sensitive or Edge and Level sensitive. In some cases, when a keyboard or button is pushed it may stay low for one second, until it is released. In Edge mode, only the edge is captured, preventing the CPU from being continuously interrupted in its low level until another falling or rising edge occurs. Edge & Level mode enables the CPU to monitor both the edge and the level when required.

Next is the Pin Enable register. For each pin that is associated with the KBI module, there is a separate bit to enable or disable it.

The Edge Control register configures the KBI for either a rising or falling edge interrupt capture. Each pin has an individual bit associated with the rising or falling edge.
### KBI and Parallel I/O Interaction

<table>
<thead>
<tr>
<th>Line #</th>
<th>PTxPEm</th>
<th>PTxDDm</th>
<th>KBIPEn</th>
<th>KBEDGn</th>
<th>Pull-up</th>
<th>Pull-down</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Pull Enable)</td>
<td>(Data Direction)</td>
<td>(KBI Pin Enable)</td>
<td>(KBI Edge Select)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>disabled</td>
<td>disabled</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>enabled</td>
<td>disabled</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>disabled</td>
<td>disabled</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>enabled</td>
<td>disabled</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>disabled</td>
<td>enabled</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>disabled</td>
<td>disabled</td>
</tr>
</tbody>
</table>

x = Don't care

The KBI module can make use of the pull-up and pull-down resistor of the port pin associated with it. Here you see a truth table for enabling and disabling the pull-up and pull-down resistor with the associated Port and KBI control bits.

Each of the available pins associated with the KBI and Parallel port has a unique bit. Note that the KBI pin assignments may span several I/O ports so that m may not always be equal to n. Please see the pinout of the specific device being used and determine which KBI pins are associated with which port pins.

The Port Pull Enable register and the Data Direction register are associated with the Parallel port while the KBI pin enable and KBI edge select registers are part of the KBI module.

In order for the pull-up or pull-down resistor to be enabled, the pull enable bit must be set.

If the KBI pin is not enabled and the Data Direction register is set to an output, the pull-up resistor is automatically disabled to prevent any excess current from being drawn.

In order for the pull-down resistor to be enabled, the Port pull enable bit, the KBI pin enable bit, and the KBIEDG bit must all be set.

When the KBI module is enabled, the Data Direction register is ignored, as indicated by the “don’t care” items in lines 4, 5, and 6.
Question

What happens if a user writes a “1” to the port Data register that is controlled by an enabled on-chip peripheral that is driving the pin low? Select the response that applies and click Done.

a. The pin drives high (logic “1”)

b. The pin will remain low (logic “0”)

c. A software interrupt will be generated

d. A reset will be generated

Done

Take a moment to complete the following question.

Correct.

If a user writes a “1” to the Port Data register that is controlled by an enabled on-chip peripheral, the pin will remain low at logic 0.
In this course you learned about the key features of the S08 port structures. You learned about individually programmable pull-ups on each pin, software-controllable slew rate output buffers, and configurable drive strength.

The second area this course covered was techniques used to reduce current draw, reduce electromagnetic radiation, and increase the ADC accuracy.

The last area this course covered was the keyboard interrupt module. You learned that it has a configurable interrupt source; it can have rising or falling edges selected for each of the pins; and it can be used for a wake-up mechanism from Stop mode, or as an interrupt mechanism in Run mode. You also learned about pin by pin enable and the single interrupt vector that is used for all eight pins associated with the KBI module.