Introduction

PURPOSE:
This course provides an overview of six modules that facilitate interfacing external devices to the i.MX21: NAND Flash Controller, Smart LCD Controller, Fast Infrared Interface, Bus Master Interface, Digital Audio Multiplexer, and Direct Memory Access.

OBJECTIVES:
- Describe features and components of the NANDF Controller
- Describe features and components of the SLCDC
- Describe features and components of the FIRI
- Describe features and components of the BMI
- Describe features and components of the AUDMUX
- Describe features and components of the DMA

CONTENT:
- 39 pages
- 6 questions

LEARNING TIME:
- 60 minutes

This course provides an overview of 6 modules that facilitate interfacing external devices to the i.MX21: the NAND Flash Memory Controller, the Smart LCD Controller, the Infrared Interface, the Bus Master Interface, the Audio Multiplexer, and Direct Memory Access.
NAND Flash Controller

(NANDF Controller)
NANDF Controller Features

• Interfaces to standard NAND Flash devices while hiding complexities of accessing NAND Flash
• Glueless interface to 8- and 16-bit NAND Flash devices
• 2 KB internal RAM buffer, used as Boot RAM for booting and buffering during normal operation
• Supports NAND Flash devices with page sizes of 512 B and 2 KB
• Error Correction Code (ECC) provides 1-bit auto correction or multi-bit error detection
• Handshake ability via interrupt pins to indicate if NAND Flash is Ready or Busy
• NAND Flash devices with densities up to 2 Gbits are supported

The NAND Flash Controller interfaces to standard NAND Flash devices while hiding the complexities of accessing NAND Flash.

It provides a glueless interface to 8- and 16-bit NAND Flash devices.

It has a 2 kilobyte internal RAM buffer, used as Boot RAM for booting operations and for buffering during normal operation.

It supports NAND Flash devices with page sizes of 512 bytes and 2 kilobytes.

The controller contains an Error Correction Code block that provides 1-bit auto correction or multi-bit error detection.

It provides handshake ability with the NAND Flash device through use of interrupt pins to indicate if the NAND Flash device is Ready or Busy.

NAND Flash devices with densities up to 2 gigabits are supported.
The diagram illustrates the error correction control block and internal ram buffer. Also shown is the NAND Flash Control block which is used to generate the control signals for the NAND Flash, the Address Control Block which is used to generate address control and generation, and other control blocks.

Blocks required for NAND Flash booting are depicted. If the BOOT signals, shown on the left, are configured for NAND Flash booting, then this information is passed to the NAND Flash controller’s BOOT block to configure it for the booting process. The BOOT Loader block is responsible for automatically loading 2 kilobytes from the NAND Flash into the 2 kilobyte internal ram buffer. Then, after exiting from the reset state, the ARM926 CPU can read the first instruction from the internal ram.

The right side of the diagram shows the external interface of the NAND Flash Controller. Click on each of the eight signals to learn more about them.
NANDF Controller BOOT Options

The upper left portion of this diagram illustrates the boot mode selection signals. The table below these signals illustrates the various system boot mode selections available to the user.

There are three options to choose from when booting from NAND Flash. The first option is booting from a 16-bit NAND Flash with a 2 kilobyte page size. The second is booting from a 16-bit NAND Flash with a 512-byte page size. The third is booting from an 8-bit NAND Flash with a 512-byte page size. Note that for proper operation, the Boot 3 signal should always remain zero or logic low.

Also note that the boot settings take place only during a system-wide power on reset. Changing the boot settings after the i.MX21 has been powered on is not recommended.

Once the NAND Flash option is chosen, this information is passed to the function multiplexing control register, specifically to the NAND Flash Memory select bit and the NAND Flash 16-bit select bit.

The NAND Flash Memory select bit is set when the 2 kilobyte page size boot option is selected, and cleared when the 512-byte page size boot option is selected.

The NAND Flash 16-bit select bit is set whenever the 16-bit NAND Flash boot option is selected and cleared when the 8-bit boot option is selected. However, if the NAND Flash device is not chosen as a boot option, these bits may be utilized by the user to indicate to the NAND Flash controller the NAND Flash data bus size and page size. The function multiplexing control register can be found in the System Control chapter of the i.MX21 reference manual.

Finally, the boot signals are passed to the NAND Flash controller where its internal logic decodes this information, and passes it to the NAND Flash controller’s boot logic where it will begin to copy 2 kilobytes of data from the first block of the NAND Flash device.
Here are NAND Flash connection examples. The left diagram depicts a NAND Flash connection scheme using a 256 mega-bit NAND flash configured as 32-meg by 8-data bits. The figure on the right illustrates a NAND Flash connection scheme using a 256-mega bit NAND Flash configured as 16-meg by 16-data bits.

Note the difference: The 8-bit NAND Flash example on the left does not use the upper data bits; that is, NAND Flash I/O 15 through 8.
Question

Which of the following statements about the NAND Flash Controller are true? Select all that apply and then click Done.

a. It contains an Error Correction Code block. *
b. It has a 4 kilobyte internal RAM buffer.
c. It offers two booting options.
d. It provides handshake ability with the NAND Flash device.*

Done

Take a moment to verify your understanding of this material about the NAND Flash Controller.

Correct.
Statements A and D are true. Statement B is not true – The NAND Flash Controller has a 2 kilobyte internal RAM buffer. Statement C is not true – the controller offers three booting options, not two.
Smart Liquid Crystal Display Controller

(SLCDC)
The traditional type of "dumb" display is in today's Personal Digital Assistants and candy-bar style phones. This display contains the X&Y drivers for the rows and columns of the display and the display itself.

Display data is stored in the i.MX21 (front and back buffers) and the display controller module or LCDC sends display data to the LCD. Display data and timing signals are sent via a parallel interface. The display data must be continually sent to the LCD module, at a rate of approximately 60 frames per second. This means that the controller is active, even if the display content is not changing!
What is a Smart Display?

Caller ID is commonly displayed on a “Smart” LCD with today’s clamshell style phones. A “smart” display contains its own frame buffer memory and display controller with X and Y Drivers.

The advantage of Smart displays is that the i.MX21 only needs to send display data when the data has changed. Additionally, the i.MX21 can enter a low-power mode, yet the display can show static content. This is because the LCD module can refresh the display from its own memory using its own display controller, potentially resulting in significant power savings, therefore a longer battery life.

The "smart" display can be connected through a serial or parallel interface. Also, the data is typically limited to 16-bits for a “smart” display, therefore not as many colors can be represented as with a “dumb” display.
Control and status registers are accessed via the IP bus. These registers are used to store information about the characteristics of the “smart” display that is connected to the system.

A DMA controller acts as an alternate master of the AHB bus. The SLCDC DMA interface requests control of the AHB in order to do 32-bit reads from system memory.

Data retrieved from system memory by the DMA controller is stored in either the image data buffer or the command data buffer.

Data is pulled from the image data and command data buffers as needed, reformatted in the bit manipulation and display interface module…

… and then sent out to the “smart” display via the SLCDC/LCDC signal multiplexer.

The SLCDC can be configured to write data to the “smart” display via a 3-line or 4-line serial interface, or an 8-bit or 16-bit parallel interface.
The SLCDC module enables the i.MX21 to connect to a “smart” display by providing the required hardware interface and software layer.

Data is transferred from the display memory buffer on the i.MX21 to the external display device via either a 3- or 4-wire serial interface, or an 8- or 16-bit parallel interface. The external device then displays the data without intervention. This frees up the i.MX21 to focus on image rendering.

The SLCDC supports a variety of displays. However, while “smart” displays have a fairly standard hardware interface, the software interface can vary significantly. To address these differences, the SLCDC contains two separate buffers: one for image data and one for control data.

The control data buffer is used to store commands that are specific to the display. The SLCDC is responsible for formatting the command and data string such that it is understood by the external display device.

The SLCDC DMA transfers data from system memory to SLCDC buffers, where it is formatted and sent to the display. The DMA, an alternate master of the AHB bus, requests control of the bus to read system memory. Typically, when the image has been rendered in system memory, the CPU triggers the SLCDC to transfer the image to the display device. This transfer is by burst DMA, which steals cycles from the CPU. Cycle-stealing is programmable. Thus, bus utilization is deterministic and can be kept within pre-defined limits.

At this time, read accesses to the “smart” display memory are not supported. Only write accesses are allowed.
SLCDC Access to *Smart* Display

- **2 methods for access**

1. **Automatic Data Transfers**
   - Preferred method
   - 3 options:
     1. Data + Control
     2. Display Data Only
     3. Control Data Only

2. **Direct Access**
   - Allows single words to be written to *Smart* display
   - Word is either 8 or 16 bits

There are two ways the SLCDC can access external “smart” displays: either automatic transfers or direct access.

Automatic data transfers is the preferred method, and can be done in one of three ways:

1. With Data plus Control transfers, the SLCDC automatically transmits control strings at appropriate times in the frame buffer data stream. Typically, the display RAM is organized into pages. After a page is filled, the display’s page address must be set to the next page. The SLCDC has counters that monitor the amount of data that has been sent to the display, therefore it knows when the page has been filled. Once a page is filled, the SLCDC inserts a control string to set the display’s page address to the next page. Display data and command strings are sent until the display’s RAM is filled.

2. With Image Data Only transfer, the SLCDC transfers the contents of the image data buffer without inserting control information; the SLCDC essentially ignores the control data buffer. Output signal LCD_RS is held high to indicate image data is being written.

3. With Control Data Only transfer, the data is a block of commands which initialize the display. Output signal LCD_RS is held low to indicate that control data is being written.

In the direct-access method, single 8- or 16-bit words are written to the display. Output signal LCD_RS indicates if it’s command or display data.
Example Output Stream from SLCDC

At the start of each page of data, three 16-bit commands are sent to the "smart" display followed by associated 16-bit display data. In this example, one page consists of a row of pixels. This cycle is repeated until the display RAM is filled. Commands are pulled from the SLCDC command data buffer, and display data is pulled from the SLCDC image data buffer.
SLCDC: Things to Remember

• Word size needs to be defined for data and commands
• Image endianness
  – Image data may be stored in 8- or 16-bit Little endian
  – IMGEND setting is used to convert image data to Big endian
  • Roll your mouse pointer over the Show Table button to learn more.

<table>
<thead>
<tr>
<th>IMGEND</th>
<th>Conversion</th>
<th>32-Bit Data in Memory</th>
<th>32-Bit Data After Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Big endian or 32-bit Little endian to Big endian</td>
<td>0x12345678</td>
<td>0x12345678</td>
</tr>
<tr>
<td>01</td>
<td>16-bit Little endian to Big endian</td>
<td>0x56781234</td>
<td>0x12345678</td>
</tr>
<tr>
<td>10</td>
<td>8-bit Little endian to Big endian</td>
<td>0x78563412</td>
<td>0x12345678</td>
</tr>
</tbody>
</table>

• Pin multiplexing
• Aborting SLCDC transfers
• Low-power operation

When using the SLCDC, there are several things that need to be remembered. First, the SLCDC supports either 8- or 16-bit transfers of display or command data. The size of the data transfer is controlled by the LCD Transfer Configuration Register. WRD_DEF_DAT defines the size of a word for all SLCDC display data registers and display data transfers, and WRD_DEF_COM defines the size of a word for all SLCDC command registers and command data transfers.

Second, display images are either big endian or 32-bit little endian format, however, the user may have an image that is stored in half word (16-bit little endian) or byte (8-bit little endian) format. To address the resulting endianness issue, the SLCDC can convert the 8- or 16-bit data to big endian format. This is achieved using a bit setting called IMGEND in the LCD Transfer Configuration Register. Roll your mouse pointer over the Show Table button to see a table which shows how the IMGEND bit setting is used to convert data from Big to Little endian format.

Third, the SLCDC pins are multiplexed with other functions on the i.MX21 and therefore need to be configured.

Fourth, SLCDC transfers that are in progress can be terminated gracefully via the ABORT bit setting. Any byte transfers that are in progress will be completed before the SLCDC goes into an idle state.

Fifth, the SLCDC does not contain registers that allow it to be programmed during low-power mode. To realize SLCDC power savings, reduce the system clock speed. However, the system clock MUST be greater than the clock to the SLCDC (HCLK_SLCDC) so that memory reads can occur fast enough for the SLCDC bit manipulation hardware.
Question

Which of the following statements about the SLCDC are true?
Select all that apply and then click Done.

a. The SLCDC supports either 8- or 16-bit transfers of display or command data. *
b. It SLCDC pins are multiplexed with other functions on the i.MX21.*
c. It The SLCDC contains two separate buffers: one for image data and one for control data. *
d. Read and write accesses to the “smart” display are allowed.

Take a moment to answer this question about the SLCDC.

Correct.
Statements A, B, and C are true. Statement D is not true – The SLCDC only supports write accesses to the “smart” display.
Fast Infrared Interface
(FIRI)
The FIRI module is capable of establishing a half-duplex link via a LED and IR detector. It supports 0.576 and 1.152 megabit per second Medium InfraRed (MIR) physical layer protocol and 4 megabit per second Fast InfraRed (FIR) physical layer protocol defined by IrDA, version 1.4. In addition, the Serial InfraRed or SIR protocol is supported via the UART modules.

The LED, IR detector, and UART pins are shared for both modules and can be controlled via GPIO, as shown.
There are two main modes of operation: hardware packet assembly and hardware packet search.

With hardware packet assembly, in intervals of 500 milliseconds or less, the transmitter must send the Serial Infrared Interaction Pulse or SIP in order to guarantee that low speed IR devices do not interfere. The pulse must be 8.7 microseconds wide with a positive phase 1.6 microseconds wide.

With hardware packet search in MIR mode, the searcher looks for the STA field. If the sequence matches, the data stream following is sent to the decoder. In FIR mode, there is a search for the PA field, then STA. If found, the decoder converts the stream to data.

Software packet assembling is not convenient to use. It is mainly for debug or testing of future IrDA standards.
You now have the opportunity to learn about the MIR and FIR packet structure and modulation. Roll your mouse pointer over each diagram to view a summary of this information.
These callouts relate to the previous slide.
**Configuration**

- FIR is muxed with GPIO Port E bits 8 & 9 as alternate functions
- Configure GPIO for IR_TXD and IR_RXD use
  - Clear Port E OCR1 bit 8 to use Ain for IR_TXD
  - Clear Port E INCONFA1 bit 8 to use Aout for IR_RXD
  - Set Port E DDIR bit 8 to set an output for IR_TXD
  - Clear Port E DDIR bit 9 to set an input for IR_RXD
  - Set Port E GIUS register bits 8-9 for GPIO function
- **Enable the clock to the module**
  - Enable the FIRI Baud Clock and IPG_CLK to the FIRI module
    - Set PCCR0 register bits 20 and 8
- **IRQ source number to enable into AITC module: FIRI_IRQ = 9**
- **DMA Requests**
  - TX_FIFO → FIRI[1] = 4
  - RX_FIFO → FIRI[0] = 5

The FIR Interface is multiplexed with GPIO Port E bit 8 and 9 as additional functions.

There are a few settings that need to be made in order to use the FIR Interface with an external device. It depends on the hardware, but some GPIOs can be used to control the mode of the external transceiver.

It is important to enable the clock to the module.

The interrupt request source number to enable into AITC module is 9.

DMA request information is shown for transmit and receive.
FIRI Programming Example

• **Set up TX path**
  1. Enable gated clocks: Baud and IPG_CLK
  2. Set oversampling and FIRI divider to get correct baud rate from SPLL
  3. With TX disabled, set mode and all areas concerning the protocol (address generation, start and stop fields, number of data, …)
  4. Enable TX
  5. Fill the TX FIFO with the number of data specified into FIRITCTR
  6. Wait for transmission completion with bit TC of FIRITSR
  7. Clear it, and it’s ready for another transmission

A programming example to set up transmit is shown. With TX disabled, setting the mode selects FIR, MIR 0.576, MIR 1.152, or packet assembling.
FIRI Software Restrictions

1. The value of FIRITCR, FIRITCTR, FIRICR registers, except TDT bits, should not be changed if TE bit is set or if transmission of current packet is not completed.

2. The value of FIRIRCR register, except RDT bits, should not be changed if RE bit is set.

3. The “write one to clear” status bits can be cleared (by software) after a period greater than chip period from a time point when it was set (by hardware). This restriction is especially true when the status bit is cleared at the beginning of interrupt routine.

4. The “clear” request of “write one to clear” status bits is accepted after a period greater than 2 chip periods.

5. The “write one to clear” status bits can not be cleared if TE/RE bit is cleared.

6. If TE bit is cleared in the middle of a transfer, the transmitter should not be enabled again if the current packet’s transmission is not completed.

7. If the RE is cleared by software, it can be set after a period greater than 4 chip periods.

The seven restrictions listed here must be adhered to.

First, the value of FIRITCR, FIRITCTR, FIRICR registers, except TDT bits, should not be changed if TE bit is set or if transmission of current packet is not completed.

The value of FIRIRCR register, except RDT bits, should not be changed if RE bit is set.

The “write one to clear” status bits can be cleared (by software) after a period greater than chip period from a time point when it was set (by hardware). This restriction is especially true when the status bit is cleared at the beginning of interrupt routine.

The “clear” request of “write one to clear” status bits is accepted after a period greater than 2 chip periods.

The “write one to clear” status bits can not be cleared if TE/RE bit is cleared.

If TE bit is cleared in the middle of a transfer, the transmitter should not be enabled again if the current packet’s transmission is not completed.

If the RE is cleared by software, it can be set after a period greater than 4 chip periods.
Which of the following statements about the FIRI are true? Select all that apply and then click Done.

a. Serial InfraRed or SIR protocol is not supported

b. There are two main modes of operation: hardware packet assembly and hardware packet search.*

c. If the RE is cleared by software, it can be set after a period greater than 2 chip periods.

d. The LED, IR detector, and UART pins are shared for both modules and can be controlled via GPIO. *

Take a moment to check your understanding of the FIRI.

Correct
Statements B and D are true. Statement A is not true – Serial InfraRed is supported. Statement C is not true – the RE can be set after a period greater than 4 chip periods.
Bus Master Interface

(BMI)
BMI Features

• The BMI provides high-speed interface between the i.MX21 and alternate bus masters in the system, such as dedicated external graphics accelerators that provide enhanced graphics for mobile gaming.
• Supports external bus master devices and external bus slave devices using memory-access timing
• Direct support for ATI Technologies Inc™ graphics processors
• 8- or 16-bit data bus interface
• RX and TX FIFOs → 16 x 32 each
• DMA support

The BMI provides high-speed interface between the i.MX21 and alternate bus masters in the system, such as dedicated external graphics accelerators that provide enhanced graphics for mobile gaming.

The BMI supports both external bus master devices and external bus slave devices using memory-access timing. Direct support is provided for ATI Technologies Inc.™ graphics processors. Either an 8-bit or 16-bit data bus interface is allowed. The receive and transmit FIFOs are 16 by 32 each. DMA support is provided.

We cannot recommend one supplier over another and in no way suggest that ATI Technologies Inc.™ is the only graphics processor supplier.
The BMI consists of the components highlighted on this diagram. Roll your mouse pointer over each highlighted component to learn about its function.
BMI Signal Configuration

<table>
<thead>
<tr>
<th>Signal</th>
<th>Configuration</th>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI_CLK_CS</td>
<td>GP22 Port A[21:16] Alternate Function</td>
<td>Clear bits 21 – 8 of GIUS_A (GPIO In-Use Register)</td>
<td>The bi-directional data bus that can be programmed to be either 8-bits or 16-bits wide.</td>
</tr>
</tbody>
</table>

You now have an opportunity to explore each of the interface signals. Simply roll your mouse pointer over each signal to learn more about it.
<table>
<thead>
<tr>
<th>Signal</th>
<th>Configuration</th>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMI_CLK_CS</td>
<td>GPIO Port [3]</td>
<td>1. Clear bit 9 of GIUS_A 2. Set bit 9 of GPR_A</td>
<td>A bi-directional signal that can be programmed as either a clock signal or a chip select signal, depending on the external master device that is connected. If an ATI Technologies Inc.™ graphics processor is utilized, this will be a clock signal. The frequency of the clock MUST be less than HCLK divided by 3.</td>
</tr>
<tr>
<td>BMI_WRITE</td>
<td>GPIO Port [23]</td>
<td>1. Clear bit 23 of GIUS_A 2. Set bit 23 of GPR_A</td>
<td>An active-low bi-directional signal to control the direction of data flow. For an external master device this signal is an input to the BMI to indicate if the data access request is a read or write. When connected to an external slave device this signal is an output from the BMI and acts as a write enable signal.</td>
</tr>
<tr>
<td>BMI_READ</td>
<td>GPIO Port [30]</td>
<td>1. Clear bit 30 of GIUS_A 2. Set bit 30 of GPR_A</td>
<td>An active-low output signal to enable external slave device data output. This signal is not used and is driven high when the BMI is connected to an external master device.</td>
</tr>
<tr>
<td>BMI_READ_REQ</td>
<td>GPIO Port [22]</td>
<td>1. Clear bit 22 of GIUS_A 2. Set bit 22 of GPR_A</td>
<td>Used to inform the external master device that data is ready for a read operation. This signal indicates that the data in the transmit FIFO is larger or equal to the transfer size of the external bus device read request.</td>
</tr>
<tr>
<td>BMI_RXF_FULL</td>
<td>GPIO Port [29]</td>
<td>1. Clear bit 29 of GIUS_A 2. Set bit 29 of GPR_A</td>
<td>A signal used to indicate that the receive FIFO has reached its watermark, or threshold value.</td>
</tr>
<tr>
<td>BMI_WAIT</td>
<td>GPIO Port [28]</td>
<td>1. Set bit 29 of GIUS_A 2. Clear bits 27–26 of ICONFA2 (GPIO Input Config Register A2)</td>
<td>An active-low signal that is only valid when the BMI is connected to a slave device and the Wait bit is set. It is a handshaking signal used by the external slave device to indicate that data is available on the bus for a read cycle or that data has been written to the slave device’s memory for a write cycle.</td>
</tr>
</tbody>
</table>
BMI Register Overview

The BMI register set consists of 5 user-accessible registers that are 32-bits wide. Except for the transmit FIFO register, each register must be word accessed. The transmit FIFO register allows both byte and half-word accesses; this is because the BMI data bus can be configured as either 8- or 16-bits wide.

**BMI Register**

- BMICTLR1 is BMI Control Register 1.
- BMICTLR2 is BMI Control Register 2.
- BMISTR is the BMI Status Register.
- BMIRXD is the BMI Receive FIFO Register.
- BMITXD is the BMI Transmit FIFO Register.

Roll your mouse pointer over each row to learn more.
Connecting BMI to Ext Devices

- **Three options exist for connecting external devices:**
  1. ATI Technologies Inc.™ GPU (Graphics Processing Unit)
     - GPU generates BMI_CLK/CS signal
     - BMI generates BMI_CLK/CS signal
  2. External Master Device
  3. External Slave Device

- **MMD_MODE_SEL, MASTER_SEL, and MMD_CLOCK_OUT bit settings in BMICRTL1 select how the device is connected**

<table>
<thead>
<tr>
<th>MMD_MODE_SEL</th>
<th>MASTER_SEL</th>
<th>MMD_CLOCK_OUT</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1            | 0          | 1             | GPU is master
|              |            |               | GPU generates BMI_CLK/CS |
| 1            | 0          | 0             | GPU is master
|              |            |               | BMI generates BMI_CLK/CS |
| 0            | 0          | N/A           | External device is master
|              |            |               | Ext device generates BMI_CLK/CS |
| 0            | 1          | N/A           | External device is slave
|              |            |               | BMI generates BMI_CLK/CS |

There are three ways to connect an external device. If the external device is an ATI Technologies Inc.™ Graphics Processing Unit (GPU), then the BMI’s special mode can be utilized. This mode contains signals and timing specifically for the GPU. In the GPU mode, either the GPU or the BMI can generate the BMI_CLK/CS signal.

If the external device is not the GPU, then it can be connected either as an external master device or an external slave device. Note that the external device MUST have bus mastering capabilities if it is to be connected to the BMI as an external master.

The table shows the bit settings required to connect the external device in various modes. Bit settings can be found in the Reference Manual; see BMI Control Register 1.

When connecting the GPU and BMI, MMD_Mode_SEL must be set to one. Master_SEL is unused. MMD_Clock_OUT selects either the GPU or the BMI to generate the BMI_CLK/CS signal.

When interfacing an external master device, bits MMD_MODE_SEL and MASTER_SEL must both be zero. Programming Master_SEL to zero configures the BMI as a slave to the external master device. In this scenario, bit MMD_Clock_OUT is ignored and the external master device generates the BMI_CLK/CS signal.

For an external slave device, bit MMD_Mode_SEL is zero and bit Master_SEL is one which configures the BMI as a master. MMD_Clock_OUT is ignored and the BMI generates the BMI_CLK/CS signal. Exclusive to this option is the ability to terminate the read/write access by either the active-low BMI_Wait signal or the pre-programmed number of wait states. This is configured via the Wait bit in BMI Control Register 1. When the Wait bit is zero, which is default, the access terminates once the programmed number of wait states has elapsed. When Wait is one, signal BMI Wait is used to indicate the end of the access. Using BMI Wait to control termination of read-write accesses can improve performance of slower slave devices.
Question

Which of the following statements about the BMI are true? Select all that apply and then click Done.

a. The BMI register set consists of 5 user-accessible registers that are 32-bits wide. *

b. An external device must have bus mastering capabilities if it is to be connected to the BMI as an external master. *

c. The BMI provides high-speed interface between the i.MX21 and alternate bus masters in the system. *

d. All registers must be word accessed.

Done

Take a moment to answer this question about the BMI.

Correct

Statements A, B, and C are true. Statement D is not true – The transmit FIFO does not need to be word accessed.
Digital Audio Multiplexer

(AUDMUX)
The AUDMUX is a digital multiplexer which supports bi-directional data transfer without the need for host processor intervention.

It offers programmable interconnects for voice, audio, and data routing between two on-chip SSI modules, and external SSI and SAP devices. Uses can be found in audio and video CODECs and modems.

The resources do not need to be hardwired and they can be shared in various configurations.

A synchronous mode, which uses a 4-wire interface, or asynchronous mode with a 6-wire interface is allowed.

Frame sync and clock direction selection add to the flexibility.
The AUDMUX allows multiple separate data flows simultaneously between ports, in either a point-to-point or point-to-multipoint configuration.

Six ports are provided on the AUDMUX. Two ports are host ports which are internal.

One port is an external host port which can interface to SAP devices.

The remaining three ports are peripheral ports with the flexibility to interface SSI, I2S, or AC97 external devices.
Data transfers bi-directionally, but only one direction is discussed here. IP Bus data is converted from parallel to serial data by SSI-1 and/or SSI-2 and brought into the AUDMUX. The AUDMUX steers the serial data to one or more of 4 ports into the external peripheral and host interface. The physical output is routed off-chip as shown at the bottom of the diagram.
Register Description

<table>
<thead>
<tr>
<th>Port No.</th>
<th>Port Name</th>
<th>Type</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SSI-1</td>
<td>Internal</td>
<td>HPCR1</td>
<td>Host Port Configuration Register</td>
</tr>
<tr>
<td>2</td>
<td>SSI-2</td>
<td>Internal</td>
<td>HPCR2</td>
<td>Host Port Configuration Register</td>
</tr>
<tr>
<td>3</td>
<td>SAP</td>
<td>External</td>
<td>HPCR3</td>
<td>Host Port Configuration Register</td>
</tr>
<tr>
<td>4</td>
<td>SSI_1</td>
<td>External</td>
<td>PPCR1</td>
<td>Peripheral Port Configuration Register</td>
</tr>
<tr>
<td>5</td>
<td>SSI_2</td>
<td>External</td>
<td>PPCR2</td>
<td>Peripheral Port Configuration Register</td>
</tr>
<tr>
<td>6</td>
<td>SSI_3</td>
<td>External</td>
<td>PPCR3</td>
<td>Peripheral Port Configuration Register</td>
</tr>
</tbody>
</table>

- Registers provide
  - Frame Sync and Clock Direction control
  - Frame Sync and Clock Source selection
  - Data Source selection
  - Synchronous or Asynchronous Mode selection
  - Transmit/Receive Data Source swapping
  - Internal Network Mode control

The AUDMUX is supported by three registers that configure the host port and three registers that configure the peripheral port.

The registers provide:
- Frame Sync and Clock Direction control
- Frame Sync and Clock Source selection
- Data Source selection
- Synchronous or Asynchronous Mode selection
- Transmit/Receive Data Source swapping
- Internal Network Mode control
Which of the following statements about the AUDMUX are true? Select all that apply and then click Done.

a. A synchronous and asynchronous modes are allowed. *

b. Resources need to be hardwired.

c. The AUDMUX is a digital multiplexer which supports bi-directional data transfer. *

d. There are five ports on the controller.

Take a moment to verify your understanding of this material about the AUDMUX.

Correct

Statements A and C are true. Statement B is not true – resources do not need to be hardwired. Statement D is not true – there are six ports.
Direct Memory Access

(DMA)
Each peripheral supporting DMA or direct memory access generates a DMA request or DMA_REQ bar signal to the DMA controller, assuming that each FIFO has a unique system address and generates a dedicated DMA_REQ bar signal to the DMA controller. For example, a USB device with 8 end-points each supporting DMA has 8 request signals.

DMA_REQ bar must be negated by the peripheral before the rising edge of DMA acknowledge or DMA_ACK; it is usually negated when the FIFO is read. The DMA controller provides DMA_ACK bar to the peripheral after a DMA burst is complete. This signal is sometimes used by the peripheral to clear status bits.

The following modules are associated with DMA: UART 1 to 4, SPI 1 to 3, SSI 1 and 2, SDHC1 and 2, FIRI, BMI, CSI, and external DMA request.
This example shows the variety of DMA features. There is a choice of one-line, FIFO, 2D-channel, or burst-data DMA transfers. All of these features are selected via DMA registers.

Channel Count register sets the number of bytes that are exchanged.

Channel Counter register is used to store the number of bytes that have been exchanged, before a stopped or failed transfer.
Consider a display of 8 pixels of 32 bits by 8 pixels of 32 bits, for a total of 64 pixels.
So $W$ equals 32 which is the display width in bytes.

We take 4 pixels of 32 bits so $X$ equals 16 bytes per row. This equals the width of the picture to be displayed.

Next, take 4 rows, so $Y$ equals 4. Number of rows for our picture is the height of the picture in pixels.

The resulting picture is using a quarter of the display area!

$W$-Size contains number of bytes that make up the display width.

$X$-Size contains number of bytes per row that define the $X$-Size of 2D memory.

$Y$-Size contains number of rows that make up the 2D memory window.
The access size is set to a size compliant with the FIFO. That is, if the FIFO is 16 bits wide, set the access size to 16 bits.

Then, burst length should be set as a multiple of the access size according to the FIFO trigger level. That is, for a FIFO of 16 bits (2 bytes) with a threshold set to 8 16-bit words, the software can burst 2 bytes minimum up to 8 by 2 bytes maximum.
DMA Chaining refers to the use of the same DMA Channel to automatically transfer a second data buffer, of perhaps a different length, between another 2 sets of Source and Destination addresses.

To achieve DMA Chaining, the source Address Register for each Channel, the destination Address Register for each Channel, and the Channel Count Register for each Channel are double buffered internally.

The Host, synchronized with the end of transfer IRQ, can update the values in these 3 registers during an ongoing DMA Transfer for the same channel in preparation for the next DMA transfer. Then, with the use of RPT & ACRPT bits, the second transfer can occur for different sources, destination addresses, and different amounts of data. However, the transfer starts automatically just after IRQ, and the user should ensure that the 3 registers have already been updated.

The chain of events is as follows: Transfer to Buffer 1 IRQ occurs. Transfer to buffer 2 has just started. Then, set address of Buffer 1 during ISR as address of Buffer 2 was set during previous ISR. Transfer to Buffer 2 IRQ occurs.
Question

Which of the following statements about the DMA are true? Select all that apply and then click Done.

a. Access size must be compliant with the FIFO. *

b. Each peripheral supporting DMA generates a DMA request to the DMA Controller. *

c. It does not support DMA Chaining.

d. It offers a choice of one-line, FIFO, 2D-channel, or burst-data DMA transfers.*

Take a moment to answer this question about the DMA.

Correct.
Statements A, B, and D are true. Statement C is not true – The DMA does support DMA Chaining.
Here is a summary of the six i.MX21 interfaces: the NAND Flash memory interface, the smart LCD interface, the infrared interface, the bus master interface, the audio multiplexor, and direct memory access.