Modern embedded systems have multiple busses.
Basic read and write transfers with no wait states

The EMC accepts single AHB transactions for reading and writing to external memory devices (EMDs). The EMC reformats single AHB transactions into the format required by the external EMD. The EMC may use multiple FCLK cycles to complete an EMD access, depending on the characteristics of the EMD and on the size of the access (word, half-word, or byte) and the width of the data bus to the EMD. An AHB access consists of an address phase and a data phase, as shown in Figure 7-2.

The EMC cannot complete EMD read and write transactions in only two FCLK cycles, so the user must configure the EMC and insert wait states in the data phase of the AHB access to complete the EMD access. Figure 7-4 shows an AHB read transaction with two wait states inserted into the data phase of the AHB transaction.

Figure 7-2 • AHB Address/Data Phase for Read Transfer

Figure 7-3 • AHB Address/Data Phase for Write Transfer

From: Actel SmartFusion Microcontroller Subsystem User's Guide
Basic read and write transfers with no wait states

![Diagram showing basic read and write transfers with no wait states. The diagram includes timelines for HCLK, HADDR[31:0], HWRITE, HRDATA[31:0], and HREADY. It also includes two phases: Address phase and Data phase. The diagram illustrates a read transfer with the address and data being transferred without any wait states.](image-url)
Read transfer with two wait states

The EMC accepts single AHB transactions for reading and writing to external memory devices (EMDs). The EMC reformats single AHB transactions into the format required by the external EMD. The EMC may use multiple FCLK cycles to complete an EMD access, depending on the characteristics of the EMD and on the size of the access (word, half-word, or byte) and the width of the data bus to the EMD. An AHB access consists of an address phase and a data phase, as shown in Figure 7-2.

The EMC cannot complete EMD read and write transactions in only two FCLK cycles, so the user must configure the EMC and insert wait states in the data phase of the AHB access to complete the EMD access.

Figure 7-4 shows an AHB read transaction with two wait states inserted into the data phase of the AHB transaction.
The EMC accepts single AHB transactions for reading and writing to external memory devices (EMDs). The EMC reformats single AHB transactions into the format required by the external EMD. The EMC may use multiple FCLK cycles to complete an EMD access, depending on the characteristics of the EMD and on the size of the access (word, half-word, or byte) and the width of the data bus to the EMD. An AHB access consists of an address phase and a data phase, as shown in Figure 7-2.

The EMC cannot complete EMD read and write transactions in only two FCLK cycles, so the user must configure the EMC and insert wait states in the data phase of the AHB access to complete the EMD access.

Figure 7-4 shows an AHB read transaction with two wait states inserted into the data phase of the AHB transaction.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4. Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30. Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Figure 3-8 Four-beat wrapping burst

Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Controlling the burst beats (length) of a transfer

- Burst of 1, 4, 8, 16, and undef
- HBURST[2:0] encodes the type
- Incremental burst

Wrapping bursts
- 4 beats x 4-byte words wrapping
- Wraps at 16 byte boundary
  - E.g. 0x34, 0x38, 0x3c, 0x30,…
- Bursts must not cross 1KB address boundaries

<table>
<thead>
<tr>
<th>HBURST[2:0]</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b000</td>
<td>SINGLE</td>
<td>Single burst</td>
</tr>
<tr>
<td>b001</td>
<td>INCR</td>
<td>Incrementing burst of undefined length</td>
</tr>
<tr>
<td>b010</td>
<td>WRAP4</td>
<td>4-beat wrapping burst</td>
</tr>
<tr>
<td>b011</td>
<td>INCR4</td>
<td>4-beat incrementing burst</td>
</tr>
<tr>
<td>b100</td>
<td>WRAP8</td>
<td>8-beat wrapping burst</td>
</tr>
<tr>
<td>b101</td>
<td>INCR8</td>
<td>8-beat incrementing burst</td>
</tr>
<tr>
<td>b110</td>
<td>WRAP16</td>
<td>16-beat wrapping burst</td>
</tr>
<tr>
<td>b111</td>
<td>INCR16</td>
<td>16-beat incrementing burst</td>
</tr>
</tbody>
</table>
Controlling the size (width) of a transfer

- HSIZE[2:0] encodes the size

- They cannot exceed the data bus width (e.g. 32-bits)

- HSIZE + HBURST determines wrapping boundary for wrapping bursts

- HSIZE must remain constant throughout a burst transfer

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>Byte</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>16</td>
<td>Halfword</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>32</td>
<td>Word</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>64</td>
<td>Doubleword</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>128</td>
<td>4-word line</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>256</td>
<td>8-word line</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>-</td>
</tr>
</tbody>
</table>
Burst Example
Burst Example

- HCLK
- HTRANS[1:0]: SEQ, BUSY, SEQ, SEQ
- HADDR[31:0]: 0x28
- HBURST[2:0]
- HREADY
- HRDATA[31:0]
Burst Example

T0  T1  T2  T3  T4  T5  T6  T7

HCLK

HTRANS[1:0]  SEQ  BUSY  SEQ  SEQ

HADDR[31:0]  0x28

HBURST[2:0]  INCR4

HREADY

HRDATA[31:0]
Burst Example

- HCLK: Sequence
- HTRANS[1:0]: SEQ, BUSY, SEQ
- HADDR[31:0]: 0x24, 0x28
- HBURST[2:0]: INCR4, INCR4, INCR4
- HREADY
- HRDATA[31:0]
Burst Example

<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTRANS[1:0]</td>
<td>SEQ</td>
<td>BUSY</td>
<td>SEQ</td>
<td>SEQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HADDR[31:0]</td>
<td>0x24</td>
<td>0x28</td>
<td>0x28</td>
<td>0x2C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HBURST[2:0]</td>
<td>INCR4</td>
<td>INCR4</td>
<td>INCR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HREADY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HRDATA[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Burst Example

- **HCLK**: Timing clock
- **HTRANS[1:0]**: Transaction type
  - T0: SEQ
  - T1: BUSY
  - T2: SEQ
  - T3: SEQ
- **HADDR[31:0]**: Address
  - T0: 0x24
  - T1: 0x28
  - T2: 0x28
  - T3: 0x2C
- **HBURST[2:0]**: Burst type
  - INCR4
  - INCR4
  - INCR4
- **HREADY**: Handshake signal
- **HRDATA[31:0]**: Data transfer
  - Data (0x28)
Burst Example

- HCLK: Timing signals
- HTRANS[1:0]: Sequential Busy
- HADDR[31:0]: Address sequence
- HBURST[2:0]: Incremental Burst
- HREADY: Ready signal
- HRDATA[31:0]: Data sequence

Timeline:
- T0: Initial sequence
- T1: Busy state
- T2: Busy state
- T3: Busy state
- T4: Sequential Busy
- T5: Sequential Busy
- T6: Sequential Busy
- T7: Sequential Busy

Data Sequence:
- Data (0x24) at T0
- Data (0x28) at T3
- Data (0x2c) at T6
Write down the memory content at the end of the execution of the following program. Assume nothing has been done to the Cortex-M3 this code is running on, and the system is coming out of a reset.

\[
\begin{align*}
\text{mov r2, #100} & \quad - r2 = 100 \\
\text{movw r1, #30} & \quad - r1 = 0x1E \\
\text{movt r1, #31} & \quad - r1 = 0x001F001E \\
\text{strh r1, [r2], #4} & \quad - [100] = 0x1E, [101] = 0x00, r2 = 104 \\
\text{str r1, [r2, 1]} & \quad - r2=105, [105]=0x1E, [106]=0, [107]=0x1F [108]=0 \\
\text{sub r2, #4} & \quad - r2=101 \\
\text{strb r1, [r2], #2} & \quad - [101]=0x1E, r2=103 \\
\text{strb r2, [r2, 3]} & \quad - [106]=0x67
\end{align*}
\]

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0x1E</td>
</tr>
<tr>
<td>101</td>
<td>0x1E</td>
</tr>
<tr>
<td>102</td>
<td>0x00</td>
</tr>
<tr>
<td>103</td>
<td>0x00</td>
</tr>
<tr>
<td>104</td>
<td>0x00</td>
</tr>
<tr>
<td>105</td>
<td>0x1E</td>
</tr>
<tr>
<td>106</td>
<td>0x67</td>
</tr>
<tr>
<td>107</td>
<td>0x1F</td>
</tr>
<tr>
<td>108</td>
<td>0x00</td>
</tr>
</tbody>
</table>
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.

Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30. Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.

Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.

Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.

Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Four-beat incrementing burst, INCR4

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.
Draw the AHB Bus timing diagram for the store instructions. Assume there are no wait states to access the memory locations.

Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.
```c
#include <stdio.h>
#include <inttypes.h>

#define REG_FOO 0x40000140

main () {
    uint32_t *reg = (uint32_t *)(REG_FOO);
    *reg -= 8;
    *reg += 2;

    printf("0x%x\n", *reg); // Prints out new value
}
```

**What happens when the blue “instructions” executes?**
8000: b480     push {r7}
8002: b083     sub sp, #12
8004: af00     add r7, sp, #0
8006: f44f 73a0 mov.w r3, #320 ; 0x140
800a: f2c4 0300 movt r3, #16384 ; 0x4000
800e: 607b     str r3, [r7, #4]
8010: 687b     ldr r3, [r7, #4]
8012: 681a     ldr r2, [r3, #0]
8014: f240 0340 movw r3, #64 ; 0x40
8018: f2c0 0301 movt r3, #1
801c: 681b     ldr r3, [r3, #0]
801e: 1ad2     subs r2, r2, r3
8020: 687b     ldr r3, [r7, #4]
8022: 601a     str r2, [r3, #0]
8024: 687b     ldr r3, [r7, #4]
8026: 681b     ldr r3, [r3, #0]
8028: f103 0202 add.w r2, r3, #2
802c: 687b     ldr r3, [r7, #4]
802e: 601a     str r2, [r3, #0]
8030: bf00     nop
8032: 4618     mov r0, r3
8034: f107 070c add.w r7, r7, #12
8038: 46bd     mov sp, r7
803a: bc80     pop {r7}
803c: 4770     bx lr
803e: bf00     nop
What happens when the blue “instructions” executes?

```c
#define REG_FOO 0x40000140

main () {
    uint32_t *reg = (uint32_t *)(REG_FOO);
    *reg -= 8;
    *reg += 2;
}
```

8000:  f44f 73a0  mov.w   r3, #320
8004:  f2c4 0300  movt   r3, #16384
#define REG_FOO 0x40000140

main () {
    uint32_t *reg = (uint32_t *)(REG_FOO);
    *reg -= 8;
    *reg += 2;
}

8000:  f4f f73a0 mov.w r3, #320 ; 0x140
8004:  f2c4 0300 movt r3, #16384 ; 0x4000

What happens when the blue “instructions” executes?
What happens when the blue “instructions” executes?

#define REG_FOO 0x40000140

main () {
    uint32_t *reg = (uint32_t *)(REG_FOO);
    *reg -= 8;
    *reg += 2;
}

8000:  f4ff 73a0 mov.w r3, #320 ; 0x140
8004:  f2c4 0300 movt r3, #16384 ; 0x4000
8008:  681a ldr r2, [r3, #0]
800a:  1f91 subs r1, r2, #6
800c:  6019 str r1, [r3, #0]
"*reg += 3" is turned into a ld, add, str sequence
"*reg += 3" is turned into a ld, add, str sequence

- Load instruction
  - A bus read operation commences
  - The CPU drives the address “reg” onto the address bus
  - The CPU indicated a read operation is in process (e.g. R/W#)
  - Some “handshaking” occurs
  - The target drives the contents of “reg” onto the data lines
  - The contents of “reg” is loaded into a CPU register (e.g. r0)
"*reg += 3" is turned into a ld, add, str sequence

• Load instruction
  - A bus read operation commences
  - The CPU drives the address “reg” onto the address bus
  - The CPU indicated a read operation is in process (e.g. R/W#)
  - Some “handshaking” occurs
  - The target drives the contents of “reg” onto the data lines
  - The contents of “reg” is loaded into a CPU register (e.g. r0)

• Sub instruction
  - An immediate add (e.g. sub r0, #6) subtracts six from this value
"*reg += 3" is turned into a ld, add, str sequence

• Load instruction
  - A bus read operation commences
  - The CPU drives the address “reg” onto the address bus
  - The CPU indicated a read operation is in process (e.g. R/W#)
  - Some “handshaking” occurs
  - The target drives the contents of “reg” onto the data lines
  - The contents of “reg” is loaded into a CPU register (e.g. r0)

• Sub instruction
  - An immediate add (e.g. sub r0, #6) subtracts six from this value

• Store instruction
  - A bus write operation commences
  - The CPU drives the address “reg” onto the address bus
  - The CPU indicated a write operation is in process (e.g. R/W#)
  - Some “handshaking” occurs
  - The CPU drives the contents of “r0” onto the data lines
  - The target stores the data value into address “reg"
Accessing memory locations from C

- Memory has an address and value
- Can equate a pointer to desired address
- Can set/get de-referenced value to change memory

```c
#define SYSREG_SOFT_RST_CR 0xE0042030

uint32_t *reg = (uint32_t *)(SYSREG_SOFT_RST_CR);

main () {
    *reg |= 0x00004000; // Reset GPIO hardware
    *reg &= ~(0x00004000);
}
```
AHB-Lite bus master/slave interface
AHB-Lite bus master/slave interface

- HSELx
- HADDR[31:0]
- HWRITE
- HSIZE[2:0]
- HBURST[2:0]
- HPROT[3:0]
- HTRANS[1:0]
- HMASTLOCK
- HWDATA[31:0]

Transfer response
- HREADY
- HRESP

Global signals
- HRESETn
- HCLK

Data
- HRDATA[31:0]

Transfer response
- HREADYOUT
- HRESP

Global signals
- HRESETn
- HCLK

Data
- HRDATA[31:0]
AHB-Lite bus master/slave interface
AHB-Lite signal definitions

- **Global signals**
  - HCLK: the bus clock source (rising-edge triggered)
  - HRESETn: the bus (and system) reset signal (active low)

- **Master out/slave in**
  - HADDR[31:0]: the 32-bit system address bus
  - HWDATA[31:0]: the system write data bus
  - Control
    - HWRITE: indicates transfer direction (Write=1, Read=0)
    - HSIZE[2:0]: indicates size of transfer (byte, halfword, or word)
    - HBURST[2:0]: indicates single or burst transfer (1, 4, 8, 16 beats)
    - HPROT[3:0]: provides protection information (e.g. I or D; user or handler)
    - HTRANS: indicates current transfer type (e.g. idle, busy, nonseq, seq)
    - HMASTLOCK: indicates a locked (atomic) transfer sequence

- **Slave out/master in**
  - HRDATA[31:0]: the slave read data bus
  - HREADY: indicates previous transfer is complete
  - HRESP: the transfer response (OKAY=0, ERROR=1)
Key to timing diagram conventions

- Timing diagrams
  - Clock
  - Stable values
  - Transitions
  - High-impedance

- Signal conventions
  - Lower case ‘n’ denote active low (e.g. RESETn)
  - Prefix ‘H’ denotes AHB
  - Prefix ‘P’ denotes APB
Wait states extend the address phase of next transfer
Wait states extend the address phase of next transfer.
Wait states extend the address phase of next transfer
APB signal definitions

- **PCLK**: the bus clock source (rising-edge triggered)
- **PRESE Tn**: the bus (and typically system) reset signal (active low)
- **PADDR**: the APB address bus (can be up to 32-bits wide)
- **PSELx**: the select line for each slave device
- **PENABLE**: indicates the 2\(^{nd}\) and subsequent cycles of an APB xfer
- **PWRITE**: indicates transfer direction (Write=H, Read=L)
- **PWDATA**: the write data bus (can be up to 32-bits wide)
- **PREAD Y**: used to extend a transfer
- **PRDATA**: the read data bus (can be up to 32-bits wide)
- **PSLVERR**: indicates a transfer error (OKAY=L, ERROR=H)
A write transfer with no wait states
A write transfer with no wait states
A write transfer with no wait states
A write transfer with no wait states
A write transfer with no wait states

Setup Phase

Access Phase
A write transfer with no wait states

Timing diagram showing:
- **PCLK**: Clock signal
- **PADDR**: Address signal
- **PWRITE**: Write signal
- **PSEL**: Select signal
- **PENABLE**: Enable signal
- **PWDATA**: Data signal
- **PREADY**: Ready signal

- **Setup Phase**: Time period from T0 to T2
- **Access Phase**: Time period from T2 to T4

Key events:
- **Addr 1**: Address being read or written to
- **Data 1**: Data being transferred
- **READ** and **WRITE** states
- **SLAVE SELECTED**
- **SLAVE** state
A write transfer with no wait states
A write transfer with no wait states

- **Setup Phase**
  - PCLK
  - PADDR
  - PWRITE
  - PSEL
  - PENABLE
  - PWDATA
  - PREADY

- **Access Phase**
  - Addr 1
  - WRITE
  - SLAVE SELECTED
  - Data 1
  - SLAVE
A write transfer with wait states
A write transfer with wait states

![Diagram showing a write transfer with wait states]

- **PCLK**: Clock signal
- **PADDR**: Address signal
- **PWRITE**: Write enable signal
- **PSEL**: Select signal
- **PENABLE**: Enable signal
- **PWDATA**: Data signal
- **PREADY**: Ready signal

**Setup Phase**: The highlighted phase indicates the setup phase where the system prepares for the write transfer.
A write transfer with wait states
A write transfer with wait states
A write transfer with wait states
A write transfer with wait states
A write transfer with wait states

- **PCLK**: Clock signal
- **PADDR**: Address signal
- **PWRITE**: Write enable signal
- **PSEL**: Select signal
- **PENABLE**: Enable signal
- **PWDATA**: Data signal
- **PREADY**: Ready signal
A write transfer with wait states
A write transfer with wait states
A write transfer with wait states
A write transfer with wait states
A read transfer with no wait states
A read transfer with no wait states

Setup phase begins with this rising edge

<table>
<thead>
<tr>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PADDR</td>
<td></td>
<td>Addr 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWRITE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PENABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRDATA</td>
<td>Data 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PREADY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Setup Phase

Access Phase
A read transfer with wait states
A read transfer with wait states

Setup phase begins with this rising edge

- T0: PCLK
- T1: PADDR
- T2: PWRITE
- T3: PSEL
- T4: PENABLE
- T5: PRDATA
- T6: PREADY

Setup Phase | Wait State | Wait State | Access Phase
APB state machine

- **IDLE**
  - Default APB state

- **SETUP**
  - When transfer required
  - PSELx is asserted
  - Only one cycle

- **ACCESS**
  - PENABLE is asserted
  - Addr, write, select, and write data remain stable
  - Stay if PREADY = L
  - Goto IDLE if PREADY = H and no more data
  - Goto SETUP if PREADY = H and more data pending
Transfers can be of four types (HTRANS[1:0])
Transfers can be of four types (HTRANS[1:0])

- **IDLE (b00)**
  - No data transfer is required
  - Slave must OKAY w/o waiting
  - Slave must ignore IDLE
Transfers can be of four types (HTRANS[1:0])

- **IDLE (b00)**
  - No data transfer is required
  - Slave must OKAY w/o waiting
  - Slave must ignore IDLE

- **BUSY (b01)**
  - Insert idle cycles in a burst
  - Burst will continue afterward
  - Address/control reflects next transfer in burst
  - Slave must OKAY w/o waiting
  - Slave must ignore BUSY
Transfers can be of four types (HTRANS[1:0])

- **IDLE (b00)**
  - No data transfer is required
  - Slave must OKAY w/o waiting
  - Slave must ignore IDLE

- **BUSY (b01)**
  - Insert idle cycles in a burst
  - Burst will continue afterward
  - Address/control reflects next transfer in burst
  - Slave must OKAY w/o waiting
  - Slave must ignore BUSY

- **NONSEQ (b10)**
  - Indicates single transfer or first transfer of a burst
  - Address/control unrelated to prior transfers
Transfers can be of four types (HTRANS[1:0])

- **IDLE (b00)**
  - No data transfer is required
  - Slave must OKAY w/o waiting
  - Slave must ignore IDLE

- **BUSY (b01)**
  - Insert idle cycles in a burst
  - Burst will continue afterward
  - Address/control reflects next transfer in burst
  - Slave must OKAY w/o waiting
  - Slave must ignore BUSY

- **NONSEQ (b10)**
  - Indicates single transfer or first transfer of a burst
  - Address/control unrelated to prior transfers

- **SEQ (b11)**
  - Remaining transfers in a burst
  - Addr = prior addr + transfer size
A four beat burst with master busy and slave wait
A four beat burst with master busy and slave wait

Master busy indicated by HTRANS[1:0]
A four beat burst with master busy and slave wait

Master busy indicated by HTRANS[1:0]

One wait state added by slave by asserting HREADY low
A four beat wrapping burst (WRAP4)
A four beat incrementing burst (INCR4)
An eight beat wrapping burst (WRAP8)
An eight beat incrementing burst (INCR8) using half-word transfers
An undefined length incrementing burst (INCR)
Because the burst is a four-beat burst of word transfers, the address wraps at 16-byte boundaries, and the transfer to address 0x3C is followed by a transfer to address 0x30.

Figure 3-9 on page 3-13 shows a read transfer using a four-beat incrementing burst, with a wait state added for the first transfer. In this case, the address does not wrap at a 16-byte boundary and the address 0x3C is followed by a transfer to address 0x40.