User Programmable Machine

Purpose:
• This module covers the User Programmable Machines of the memory controller and how they interface to a wide range of memory devices

Objectives:
• Understand how to use the UPMs with register programming and interface timing examples.

Contents:
• This section covers the various entries of the UPM block and the timing associated with them. The registers are covered in detail with a programming procedure at the end of the section.

Learning Time:
• There are 9 pages and 5 reference pages which will take approximately 17 minutes.

This module covers the programmable machine used in timing coordination for SDRAM devices.

To understand the inner workings of the UPM block, as well as how to apply it in provided examples.

This section covers the various entries of the UPM block and the timing associated with them. The registers are covered in detail with a programming procedure at the end of the section.

There are 9 pages and 5 reference pages which will take approximately 17 minutes.
The basis of the UPM is a block of ram containing 64 entries of 32-bits. Each entry relates to a single clock cycle, and it defines what signals and levels are asserted during that clock cycle. The RAM is divided into distinct blocks related to specific functions. The first block contains 8 entries and is used for single beat read cycles. The next block contains 16 entries and is used for burst read cycles. There are two similar blocks for write cycles. The periodic timer block contains 12 entries an is used for timing related signals such as DRAM refresh.

Finally, there is a four entry block to determine what happens to the signals in the case of an exception related to bus cycles, such as a bus error. These entries control a hardware interface to the signals which has two clock inputs. The input clock to the memory controller is copied and shifted by 90°, producing two clock inputs out of phase. The result produces four clock edges within one input clock period, from which four independent timing periods are generated. Using these, it’s possible to generate signals to a quarter clock cycle resolution.

These controls are used for the chip select pins, byte select pins and six general purpose pins. When a bus cycle is generated that matches the bank that selects a UPM to control the signals, the controller automatically jumps the to RAM block associated with the type of access, either read or write, single or burst, and starts producing the outputs defined by the RAM entries. It’s important to realize that the signals produced are what the user programs into the RAM. The user is totally in control. The only interference from the controller is relating those signals to the required outputs.

For instance, the user does not define the chip select pin affected, that is chosen by the bank, that is affected by this bus cycle. Similarly, the byte select pin affected will be chosen by the size of transfer, port size, and low order address bits. Although the pins associated with the transfers are called chip select and byte select, that is not what they have to be used for. They are simply pins controlled by the UPM and the user can use them in any way that is useful. The chip select pins could be used for the row address strobe for DRAM and the byte select for the column address.
This diagram shows the relationship between the input clock and the timing controls. The timing pulses, T1 through 4, are generated at quarter clock cycle intervals. T1 controls CST1 and BST1, T2 controls CST2 and BST2, T3 controls CST3 and BST3 and T4 controls CST4 and BST4.

The general purpose lines are not quite so finely controlled. T1 controls G1T1 and G2T1 for a half cycle and T3 controls G1T3 and G2T3 for a half cycle.
Each RAM entry consists of 32 bits, and there are 32 entries. Each entry defines signal conditions during one input clock cycle, related to one of four timing pulses. This diagram shows bits 0-23 of an entry. The first four bits define the state of the chip select line during each one of the timing pulses. Simply defining the value generates that level on the signal for each quarter clock, providing the possibility of generating a square wave at twice the frequency of the input clock.

If CST1 is set, then for the first quarter clock cycle the chip select pin will go high. If CST2 is also set, then the pin will remain high for the next quarter cycle. If CST3 and 4 were clear, then the pin will go low for the last half cycle, resulting in the pin having a waveform the same as the clock. In this way, the pin could be high for the whole clock cycle, low for the whole clock cycle or any combination of pulses to a quarter clock resolution.

BST 1-3 offer the same options for the byte select lines. The general purpose lines one through four have the same possibilities except the controls are only to a half clock resolution. The timing of the control relates to T1 and T3. For example, if G1T1 is set, then from the start of T3 general purpose line one will go high and remain so until the start of T3. Then whatever the state of G1T3 is will be reflected on the pin.

General purpose line zero is slightly different, having the option to be controlled by the state of a control bit in the machine mode register, field G0CL. The first half of the clock cycle is controlled by G0L0 and 1 as a pair, and the second half by G0H0 and 1. If the pair have the value 1, 0 then the pin will be low; if they have the value 1, one then it will be high, and 0, 0 causes the pin to be controlled by the machine mode register.

Notice that GT4 pins are shared with a delay timer and wait enable. The pin function is selected by the machine mode register. The signals generated by the UPM may require several cycles and some of them may be identical. In that case consecutive identical signals do not need to be re-defined as the redo field enables the current entry to be repeated two, three, or four times.
### Fourth Byte of UPM RAM entry (1 of 64)

<table>
<thead>
<tr>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOOP</td>
<td>EXEN</td>
<td>AMX0</td>
<td>AMX1</td>
<td>NA</td>
<td>UTA</td>
<td>TOTD</td>
<td>LAST</td>
</tr>
</tbody>
</table>

- **START/END Loop Pattern**: Count defined in MxMR
- **A28-A31 Incremented @ GCLK1 as per Port**
- **A0_A31**:
  - 00 = normal
  - 10 = mux'd as per MxMR/AMx
  - 11 = as per MAR
- **Exception Enabled**: Branch to Exc. Pattern after current cycle if TEA or RESET asserted
- **Disable Timer Enabled**: No new access to bank till timer expired
- **UPM Service Complete**
- **TA state @ GCLK2**

**Reference Manual section 11.6.4.1**

The fourth byte of the entry provides control for the remainder of the signals, which may be necessary for the bus cycle. It may be necessary to repeat a pattern several times. In that case, the loop bit is used. For the required pattern, the loop bit is set in the first entry and the last entry of the pattern.

For the first clock cycle, the controller generates outputs as defined by that first entry. At the end of the clock cycle, the controller steps to the next entry, unless the redo bit is set, and then performs the outputs as defined in the second entry. This process repeats down the entries.

The first time the controller meets an entry with the loop bit set it remembers that entry but continues on until it reaches a second entry with the loop bit set. When it has completed that entry, it loops back to the previous entry with the loop bit set. This will be repeated for a count defined in the appropriate field of the machine mode register. Once the count has expired, the controller then continues down the entries.

EXEN enables exceptions. When an external device is accessed, it could generate a transfer error signal at any time. The processor will go to exception processing, but the appropriate reaction on the pins may not be what is present at the time. This decides when in the process the controller reacts to the exception and jumps to the entries for this occurrence.

If the controller is handling DRAM, or something similar it will need to control address multiplexing. AMX0 and 1 define if address multiplexing takes place and, if so, whether it is controlled by the machine mode register or the memory address register. For burst accesses, the lower order addresses must be incremented for each transfer. NA determines if the address is incremented.

All transfers must result in a transfer acknowledge signal back to the controller or processor. UTA determines which entry in the cycle generates the TA signal. There is often a requirement for a minimum time between successive transfers to the same memory bank, and so there is a timer with it's period defined in the machine mode register. TOTD defines if the timer is used or not.

Finally, the controller automatically increments through the RAM entries to generate the necessary control for the bus cycle to access this bank of memory. Clearly it should not continue indefinitely through the complete block of ram, so the last bit identifies the last entry of the pattern. When the controller reaches the entry with that bit set, it completes the control for that clock cycle and then terminates. That is the end of the bus cycle. For any further bus cycles to this memory bank, the controller jumps to the start of the appropriate block and repeats the process.
This example shows the timing diagram and basic entries for the control of a burst read of page mode DRAM.

Only the entries required for this example are shown. To show the values the first and last byte of the RAM entries are indicated vertically, and each required entry is defined vertically and aligned to the timing it relates to. These values will start at RAM entry number eight, the first for the burst read cycle.

For the transfer, during the first clock cycle the row address is asserted, followed by four column addresses, each of which will be asserted for two clock cycles. What the controller must do is generate the row address strobe, column address strobe, the transfer acknowledge, and define when the address should be multiplexed and incremented. The chip select is to be used for the row address strobe, and byte select lines for the column.

In the first clock, RAS is asserted and so all four CST bits are clear so that the signal goes low and remains low for the complete clock cycle. During the same clock cycle the CAS must remain high and so all four BST bits are set. The address is to be multiplexed for the row and so AMX is 1, 0. All other bits should be low.

During the next clock cycle, RAS remains low and so do the associated bits. The column address strobe must go low a quarter cycle into the period and so the first bit is set and the remaining bits for BST are clear. The address is not to be multiplexed as it is a column address and so AMX is 0, 0.

UTA is set because the transfer acknowledge must be asserted. The next two bus cycles are to be the same as this column, and so the loop bit is set and the counter in the machine mode register initialized accordingly. In the next cycle, RAS remains low and CAS negates half way through the cycle, and so BST1 and two are clear and three and four are set. TA will be negated so UTA is clear and the address should be incremented for the next access and so NA is set.

This is the end of this cycle and so the loop bit is set. In that case, this pair of entries will be repeated the number of times defined in the machine mode register, in this case twice more. If an error should occur, then the end of the cycle is an appropriate time to handle the exception, so EXEN is set. Although the last cycle required is almost identical, the same entries cannot be repeated because RAS must be negated.

In this case, the only difference for the first entry for the last cycle is that the loop bit is not set. In the last entry, CST3 and 4 are set to drive RAS high for the last half cycle, again the loop bit is clear and the last bit is set to define this as the last entry. If the disable time is needed to ensure a delay before another access to this bank, then TODT is set. So, for this burst access, only five entries in the RAM are required.
This and the next page identifies the registers required for use with the UPM. The base register determines the start address of the bank and the machine to be used. The associated option register determines the size of the bank and some of the access attributes.

If an address generated lies within the boundaries of these two registers, and the valid bit is set in the base register then the controller defined in the base register, will control the transfer. The machine mode register defines the configuration of the controlling machine. When you decide to program the UPM RAM, you may have a problem finding it. That’s because it is not directly accessible to the user.

The RAM is programmed by loading the required entry data into the MDR and performing a write command, defined in the machine mode register. One of the options for address multiplexing is to drive the address from the memory address register instead of the multiplexed requesting address. To view the registers, click on the one of interest.
**Registers continued**

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPTPR</td>
<td>Memory Refresh Timer Prescaler Register - defines the period of the refresh timer clock.</td>
</tr>
<tr>
<td>PURT</td>
<td>60x Bus-Assigned Refresh Timer for the UPM - Defines the refresh period for DRAM on the 60x bus</td>
</tr>
<tr>
<td>LURT</td>
<td>Local Bus-Assigned Refresh Timer for the UPM - Defines the refresh period for DRAM on the Local bus</td>
</tr>
</tbody>
</table>

PURT and LURT provide the refresh rate as an 8-bit value for either the 60x bus or local bus, if the UPM is to be used for DRAM or something similar. And if refresh is required, then the refresh counter clock is defined using a prescaler to the input clock. The prescaler value is defined in the MPTPR. To view the MPTPR, click on it.
This shows the basic programming procedure for the UPM. There is no specific order except that the ram must be programmed by loading the MDR followed by a write command defined in the MXMR and the final step is to set the valid bit in the Base register which enables the controller and bank. To view the registers, click on the one of interest. That completes the user programmable machine section.