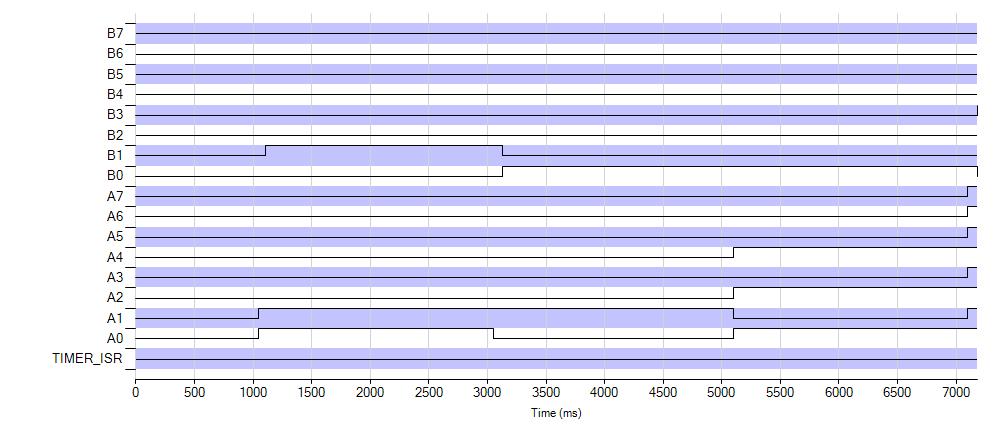
Souradeep Bhattacharya

EE120B HW 1

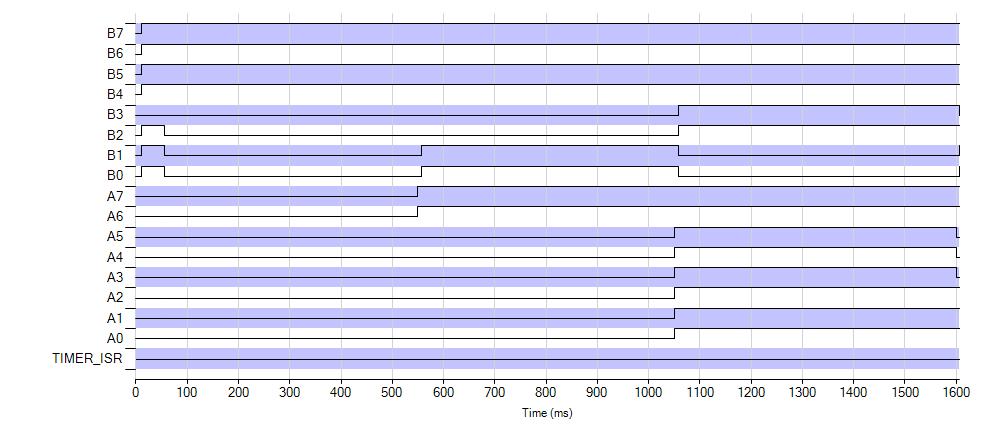
# Problem 1

|  |
| --- |
| #include "rims.h"  void main**()**  **{**  **while** **(**1**)**  **{**  char input **=** A**;**  char count **=** 0**;**  char maxCount **=** 0**;**  char bitMask **=** 1**;**  char i **=** 0**;**  **for(;** i **<** 8**;** i**++)**  **{**  **if((**input **&** bitMask**)** **==** bitMask**)**  **{**  count**++;**  **}**  **else**  **{**  **if** **(**count **>** maxCount**)**  **{**  maxCount **=** count**;**  **}**  count = 0;  }  bitMask = bitMask << 1;  }  if(count > maxCount)  {  maxCount = count;  }  B = maxCount;  }  } |

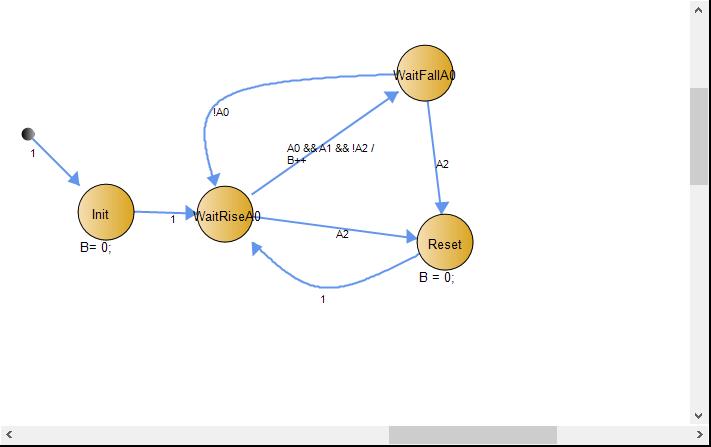


# Problem 2

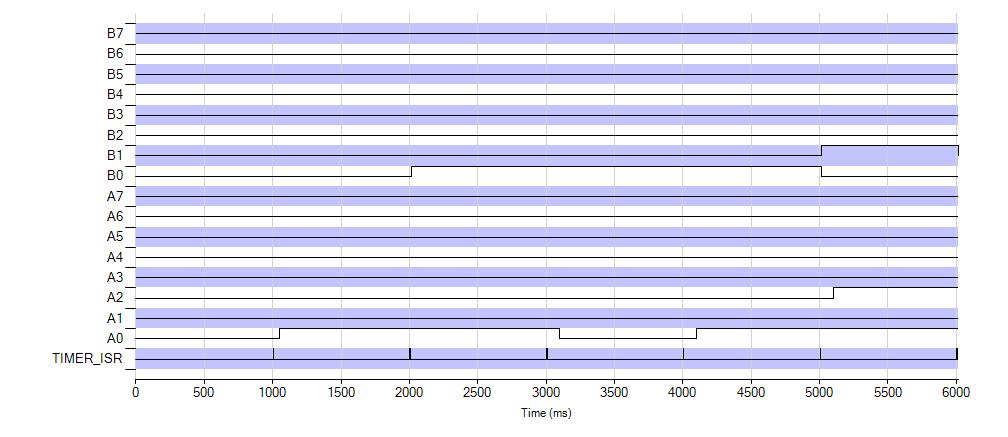
|  |
| --- |
| #include "rims.h"  void main**()**  **{**  **while** **(**1**)** **{**    char sum **=** 0**;**  char A1 **=** A **&** 0x03**;**  char A2 **=** **(**A **&** **(**0x03 **<<** 2**))** **>>** 2**;**  char A3 **=** **(**A **&** **(**0x03 **<<** 4**))** **>>** 4**;**  char A4 **=** **(**A **&** **(**0x03 **<<** 6**))** **>>** 6**;**  sum **=** A1 **+** A2 **+** A3 **+** A4**;**  B **=** 0xF0 **|** sum**;**  **}**  **}** |



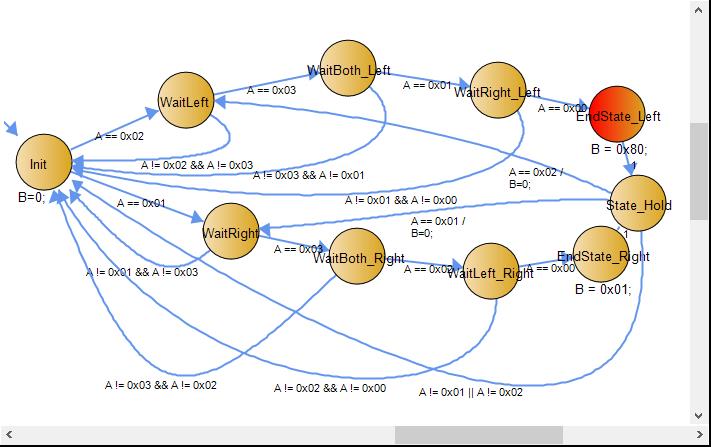
# Problem 3



|  |
| --- |
| #include "rims.h"  enum SM1\_States **{** SM1\_Init**,** SM1\_WaitRiseA0**,** SM1\_WaitFallA0**,** SM1\_Reset **}** SM1\_State**;**  TickFct\_State\_machine\_1**()** **{**  **switch(**SM1\_State**)** **{** // Transitions  **case** **-**1**:**  SM1\_State **=** SM1\_Init**;**  **break;**  **case** SM1\_Init**:**  **if** **(**1**)** **{**  SM1\_State **=** SM1\_WaitRiseA0**;**  **}**  **break;**  **case** SM1\_WaitRiseA0**:**  **if** **(**A0 **&&** A1 **&&** **!**A2**)** **{**  SM1\_State **=** SM1\_WaitFallA0**;**  B**++;**  **}**  **else** **if** **(**A2**)** **{**  SM1\_State **=** SM1\_Reset**;**  **}**  **break;**  **case** SM1\_WaitFallA0**:**  **if** **(**A2**)** **{**  SM1\_State **=** SM1\_Reset**;**  **}**  **else** **if** **(!**A0**)** **{**  SM1\_State **=** SM1\_WaitRiseA0**;**  **}**  **break;**  **case** SM1\_Reset**:**  **if** **(**1**)** **{**  SM1\_State **=** SM1\_WaitRiseA0**;**  **}**  **break;**  **default:**  SM1\_State **=** SM1\_Init**;**  **}** // Transitions  **switch(**SM1\_State**)** **{** // State actions  **case** SM1\_Init**:**  B**=** 0**;**  **break;**  **case** SM1\_WaitRiseA0**:**  **break;**  **case** SM1\_WaitFallA0**:**  **break;**  **case** SM1\_Reset**:**  B **=** 0**;**  **break;**  **default:** // ADD default behaviour below  **break;**  **}** // State actions  **}**  int main**()** **{**  SM1\_State **=** **-**1**;** // Initial state  B **=** 0**;** // Init outputs  **while(**1**)** **{**  TickFct\_State\_machine\_1**();**  **}** // while (1)  **return** 0**;**  **}** // Main |



# Problem 4



|  |
| --- |
| #include "rims.h"  enum SM\_States **{** SM\_Init**,** SM\_WaitLeft**,** SM\_WaitRight**,** SM\_WaitBoth\_Right**,** SM\_WaitBoth\_Left**,** SM\_WaitRight\_Left**,** SM\_WaitLeft\_Right**,** SM\_EndState\_Left**,** SM\_EndState\_Right**,** SM\_State\_Hold **}** SM\_State**;**  TickFct\_State\_machine\_1**()** **{**  **switch(**SM\_State**)** **{** // Transitions  **case** **-**1**:**  SM\_State **=** SM\_Init**;**  **break;**  **case** SM\_Init**:**  **if** **(**A **==** 0x02**)** **{**  SM\_State **=** SM\_WaitLeft**;**  **}**  **else** **if** **(**A **==** 0x01**)** **{**  SM\_State **=** SM\_WaitRight**;**  **}**  **break;**  **case** SM\_WaitLeft**:**  **if** **(**A **==** 0x03**)** **{**  SM\_State **=** SM\_WaitBoth\_Left**;**  **}**  **else** **if** **(**A **!=** 0x02 **&&** A **!=** 0x03**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **case** SM\_WaitRight**:**  **if** **(**A **==** 0x03**)** **{**  SM\_State **=** SM\_WaitBoth\_Right**;**  **}**  **else** **if** **(**A **!=** 0x01 **&&** A **!=** 0x03**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **case** SM\_WaitBoth\_Right**:**  **if** **(**A **==** 0x02**)** **{**  SM\_State **=** SM\_WaitLeft\_Right**;**  **}**  **else** **if** **(** A **!=** 0x03 **&&** A **!=** 0x02**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **case** SM\_WaitBoth\_Left**:**  **if** **(**A **==** 0x01**)** **{**  SM\_State **=** SM\_WaitRight\_Left**;**  **}**  **else** **if** **(**A **!=** 0x03 **&&** A **!=** 0x01**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **case** SM\_WaitRight\_Left**:**  **if** **(**A **==** 0x00**)** **{**  SM\_State **=** SM\_EndState\_Left**;**  **}**  **else** **if** **(**A **!=** 0x01 **&&** A **!=** 0x00**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **case** SM\_WaitLeft\_Right**:**  **if** **(**A **==** 0x00**)** **{**  SM\_State **=** SM\_EndState\_Right**;**  **}**  **else** **if** **(**A **!=** 0x02 **&&** A **!=** 0x00**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **case** SM\_EndState\_Left**:**  **if** **(**0**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **else** **if** **(**1**)** **{**  SM\_State **=** SM\_State\_Hold**;**  **}**  **break;**  **case** SM\_EndState\_Right**:**  **if** **(**0**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **else** **if** **(**1**)** **{**  SM\_State **=** SM\_State\_Hold**;**  **}**  **break;**  **case** SM\_State\_Hold**:**  **if** **(**A **==** 0x01**)** **{**  SM\_State **=** SM\_WaitRight**;**  **}**  **else** **if** **(**A **==** 0x02**)** **{**  SM\_State **=** SM\_WaitLeft**;**  **}**  **else** **if** **(**A **!=** 0x01 **||** A **!=** 0x02**)** **{**  SM\_State **=** SM\_Init**;**  **}**  **break;**  **default:**  SM\_State **=** SM\_Init**;**  **}** // Transitions  **switch(**SM\_State**)** **{** // State actions  **case** SM\_Init**:**  B**=**0**;**  **break;**  **case** SM\_WaitLeft**:**  **break;**  **case** SM\_WaitRight**:**  **break;**  **case** SM\_WaitBoth\_Right**:**  **break;**  **case** SM\_WaitBoth\_Left**:**  **break;**  **case** SM\_WaitRight\_Left**:**  **break;**  **case** SM\_WaitLeft\_Right**:**  **break;**  **case** SM\_EndState\_Left**:**  B **=** 0x80**;**  **break;**  **case** SM\_EndState\_Right**:**  B **=** 0x01**;**  **break;**  **case** SM\_State\_Hold**:**  **break;**  **default:** // ADD default behaviour below  **break;**  **}** // State actions  **}**  int main**()** **{**    SM\_State **=** **-**1**;** // Initial state  B **=** 0**;** // Init outputs  **while(**1**)** **{**  TickFct\_State\_machine\_1**();**  **}** // while (1)  **}** // Main |

