# 8. INTERTASK COMMUNICATION

## 8.1 Introduction

It is possible for tasks to communicate with each other within the system, by issuing 'messages'. These messages may be directed to a specific task or may be general, i.e. issued to all or any of the other tasks. Similarily, tasks may request a message from a specific task, or from any task that has issued a message.

Two queues are maintained in the system for unaddressed message requests, one for input (read) messages, and one for output (write) messages. Only one queue may have entries at one point in time. When both queues have an entry the message transfer takes place and both items are removed from the queue.

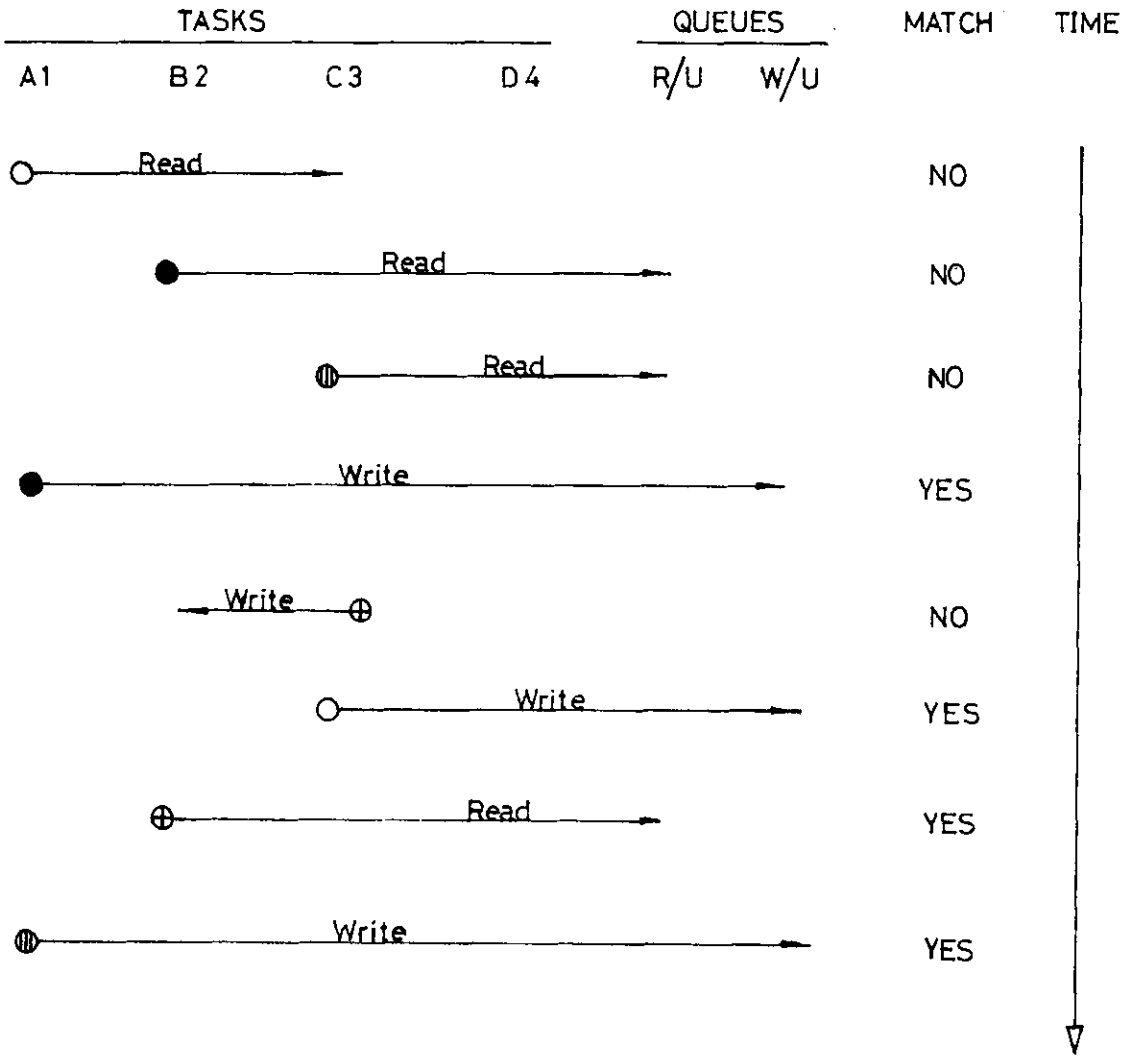
This queueing sytem works on the first in first out (FIFO) principal, with one exception that will be seen later in this section. The simplest case is as follows:-

Task A issues an unaddressed write command. This request is placed in the unaddressed write queue, until such a time as Task B issues a read; at this point the two requests are satisfied, and the queues cleared. Task A would then be able to resume, assuming the instruction had the wait bit set.

In addition, a task may issue a specific read for a message from another task, and this may result in the reception of a message specifically directed to the reading task, or one from the queue of unaddressed writes, providing the message in the latter case was issued by the task from which the current task is now requesting a message, and no matching addressed request exists.

Furthermore, a task may issue a specific write directed at another task, and in this case, if no read is oustanding from that task in either of the queues, this request is queued until such a time as the specified task issues a read directed at the task sending the message. The .NW and WAIT functions apply to intertask I/O in the same way as for conventional I/O.

# INTERTASK COMMUNICATION



### 8.1.1 Unaddressed read and write

These use the standard read and write commands, the data set directive being the TOSS file code defined for intertask communication. Examples of these instructions are shown below:-

```
READ   DSIC,MESBUF,SIZE
WRITE  DSIC,MESBUF,SIZE
```

### 8.1.2 Addressed read and write

These use the same read and write commands as random I/O, and again the data set directive will be using the TOSS file code defined for intertask communication, and in place of the record number will be the appropriate task identifier. If an addressed read is issued, then the unaddressed queue will first be checked to see if the task specified has already issued an unaddressed write, if none is found then the addressed queue will be searched, and if a match was not found then the task will be suspended until the read can be satisfied. Examples of these instructions are shown below:-

```
RREAD   DSIC,MESBUF,SIZE,TASKID
RWRITE  DSIC,MESBUF,SIZE,TASKID
```

### 8.1.3 Examples of intertask communication

Four tasks exist in the system; A1, B2, C3 and D4.

| Action | Result |
|---|---|
| A1 issues a READ ADDRESSED to C3 | Request queued in C3 |
| B2 issues a READ UNADDRESSED | Request in R/U queue |
| C3 issues a READ UNADDRESSED | Request in R/U queue |
| A1 issues a WRITE UNADDRESSED | Matched to 1st in R/U queue<br>message passed to B2 |
| C3 issues a WRITE ADDRESSED to B2 | Request queued on B2 |
| C3 issues a WRITE UNADDRESSED | Matched to A1 R/A on C3<br>message passed to A1 |
| B2 issues a READ UNADDRESSED | Matched to W/A on C3<br>message passed to C3 |
| A1 issues a WRITE UNADDRESSED | Matched to 1st in R/U queue<br>message passed to C3 |

| Keyword | Page in M04 |
|---|---|
| READ | 1.4.137 |
| RREAD | 1.4.140 |
| RWRITE | 1.4.143 |
| WRITE | 1.4.169 |