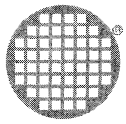


MMS2 User Guide

Contents

1 Introduction

This guide explains how to set up and run the MMS (Module Motherboard Software) version 4.0. It has been written on the assumption that the user has a TDS such as the IMS D700, and is familiar with its use. It also assumes that the user is familiar with the C004 control architecture used on INMOS module motherboards (see Motherboard Architecture Manual).

The MMS is a program (or more correctly an 'EXE') which runs under the INMOS TDS. It is designed to help users to easily set up the IMS C004 'soft' connections on module motherboards or cascades of motherboards.

To use the MMS one or more motherboards together with a number of modules are needed together with a description of the hardware and the connections that need to be made between the modules on the board.

When the MMS is run it provides interactive control of a motherboard or system of motherboards, presenting a friendly menu-driven user interface. It offers a number of facilities: setting the C004s as specified in the source file, manual C004 command input, special network mapper programs, etc.

2 Using the MMS.

2.1 Installing the MMS

To install MMS put the MMS disk into floppy drive **a:** and type

```
a:install
```

This batch file creates the directory **mms2** on the Winchester and copies the contents of the disk into this directory. The archive program **arc.exe**, is then used to extract the data from the archive file. This directory now contains a TDS **toplevel.top** file which contains the CODE EXE of MMS, together with some example source folds for the B008 and B012 module motherboards.

2.2 Getting Started

In the rest of this manual it is assumed that the motherboards in use have been set up, and that you are familiar with the user guides for them.

To be able to configure the links connecting the C004s on the motherboards the MMS reads a fold bundle containing two filed folds, known as the 'softwire' and 'hardwire' folds. The first of these contains a description of the connections that the user wants to make using the C004 links. The second contains a description of the hardware configuration of the boards being used.

The 'hardwire' fold is needed so that MMS is able to determine what connections it is possible to make; it contains information on such things as the number of C004s, number of module slots, and the connections between them. Once this description has been set up no changes will have to be made unless physical changes are made to the motherboard or system of motherboards. If you are using a single B008 or B012 there should not be any need to understand the required information in great detail as the supplied hardware description folds for these boards can be used without modification.

The 'softwire' fold is needed to specify both connections from module to module and from module to edge on a motherboard. Unlike the 'hardwire' fold the 'softwire' fold will tend to vary from one application to another.

A typical fold bundle will look similar to this:

```
{{{ Motherboard configuration bundle
...F softwire fold
...F hardwire fold
}}}
```

You should read Chapter 4 'The Softwire Fold' and study the examples supplied with MMS before attempting to run MMS or trying to set up your own 'softwire' description. To get going initially it would probably be easiest to try modifying a copy of one of the example foldsets provided.

2.3 Using the MMS

Then [GET] the MMS CODE EXE and [RUN] while pointing at the source fold bundle.

After pressing [RUN] the MMS will display a menu screen and prompt '**key command**'. At this point the user can enter any one of the command codes listed on the menu, including '**h**' for help and '**q**' to quit.

These codes are :

H — Help

Q — Quit

S — Set C004 links

C — Check source folds

D — Diagnostics on/off

N — Generate a network map

M — Manual command entry

I — Change link numbers

V — View source fold

R — Reset subsystem

I — Initialise C004s

B — Make a bootable file

O — Make an occam table

They perform the following functions :

HELP

Keying **H** allows the user to call up a number of help screens. The help screen called by keying **H** displays generally useful system information, including the implementation limits on numbers of C004s, T212s, slots etc. The MMS version number is also displayed in this help screen.

SET

Keying **S** performs the C004 setting as described in the source fold. To carry out this command the MMS first reads the hardwire fold, checks it for errors and builds up an internal representation of the system hardware. Any errors detected are reported and the command abandoned.

The reporting of errors detected in both the hardwire fold and the softwire fold is described in a later section.

Next the MMS resets the motherboard(s) and boots the config pipe using a special network mapper. If this mapper reports finding a number of T212s different to that defined in the hardwire fold it reports the fact and abandons the command.

Finally the MMS reads the softwire fold, generates and sends the configuration command sequences that it requires. Again, if any errors are detected they are reported and the command abandoned.

As well as reporting errors the **set** command will also report any connections between edges on any one board which the user needs to wire together to complete connections specified in the softwire fold.

CHECK

Keying **C** checks the source fold bundle for errors. This command is almost exactly the same in effect as the **SET** command, with the difference that no boot code or commands are actually sent to the config pipe.

DIAGNOSTICS ON/OFF

Keying **D** toggles 'diagnostic mode' on and off. With diagnostic mode on any command sequences which are generated for the config pipe (during a check, set, bootable file or occam table command) are displayed on the screen. Apart from providing the user with this extra information, diagnostic mode has no effect on the operation of the system.

NETWORK MAPPER

Sends a network mapper into the motherboard. This mapper differs from other mappers in that it explores links 1 and 2 first. The result of this is that, as long as modules are hardwired sequentially in an unbroken link 1-2 pipeline, their network map IDs will be the same as their order in this pipeline. So for example, an INMOS motherboard with modules in slots 2,3,4 and 5 would report these as having IDs of 0,1,2 and 3 respectively.

MANUAL COMMAND ENTRY

When **M** is keyed the MMS first reads the hardwire fold (in the same way as **set** etc) in order to build up an internal representation of the config pipeline. If all has gone well the user is then prompted to enter command sequences for the C004s.

These sequences are of the same form as those generated automatically : a C004 ID followed by a C004 command, followed by the parameters required by this command (if any). The C004 ID required is the C004 ID which the user has referred to in the hardwire fold. The command code can be any of the (byte) command codes which the C004 responds to (0 to connect one hard channel to another, 1 to connect links, 4 to reset).

CHANGE LINK NUMBERS

TDS transputer which the MMS expects to be connected to the 'config-up' and the 'pipe-head' links of the motherboard. The default settings are link 1 for the config-up and link 2 for the pipe-head. This option is only required if you wish to connect your boards to a host system in a non-standard way.

VIEW

Keying **V** allows the user to view the source folds from within MMS. It prompts for a source line number, and for whether it is in the hardwire or softwires fold. It then displays that line, together with the two lines preceding and following it.

RESET

Keying **R** asserts the subsystem reset on the host TDS transputer for a millisecond. This allows the user to 'manually' reset a motherboard (or other system).

INITIALISE

Keying **I** will cause all the C004s in the system to be reset. To do this the MMS must first read the the hardwire fold, then send a 'reset' byte to every C004 it mentions.

MAKE BOOTABLE FILE

When **B** is keyed the MMS checks the source folds and generates the commands (in the same way as the set command), instead of sending these to the config pipeline, they are written to a DOS file together with a program which can configure the pipeline. This file is bootable from the server in the same way as a bootable code program file (see the TDS manual for more information about code programs). This allows the links to be set up from the DOS command line without running the TDS

MAKE OCCAM TABLE

When **O** is keyed the MMS performs the same actions as for the set command except that the resulting commands are written to a filed fold in the form of an occam table. This table can be booted down a link from within an EXE or PROGRAM to configure the network.

3 The Hardwire Fold

3.1 Introduction

A fuller definition of the hardware description language, HL1, and its use is included in a later appendix. However the following section should be sufficient to allow the user at least to start using the supplied examples.

3.2 Language format

Spaces, line feeds and indentation can be inserted anywhere in a hardwire description, other than in the middle of a word or number. Where spaces and line feeds appear in this description they are only used to improve readability.

Comments are inserted using the same syntax as occam : comments can be either be put in a **COMMENT** fold, or on a line following a double hyphen "--".

3.3 General approach

The hardwire fold describes hardware systems composed of slots, C004s, links taken to edges connectors (**EDGES**), T212s and C004s. Its function is only to state which links on slots, C004s and edges are hardwired to each other.

The C004s are assumed to be controlled by a T212/C004 config pipeline described in the module motherboard background document, available from INMOS.

Each slot, T212, C004 and EDGE in the system is assigned an identity number which is referred to throughout the system description.

Any connection must be mentioned only once, for example having described a :

```
'SLOT a,LINK 2 TO SLOT b,LINK 1'
```

then describing a

```
'SLOT b,LINK 1 TO SLOT a,LINK 2'
```

connection later would be unnecessary and illegal.

3.4 Using HL1

The following is a full hardwire description of the module motherboard pictured below :

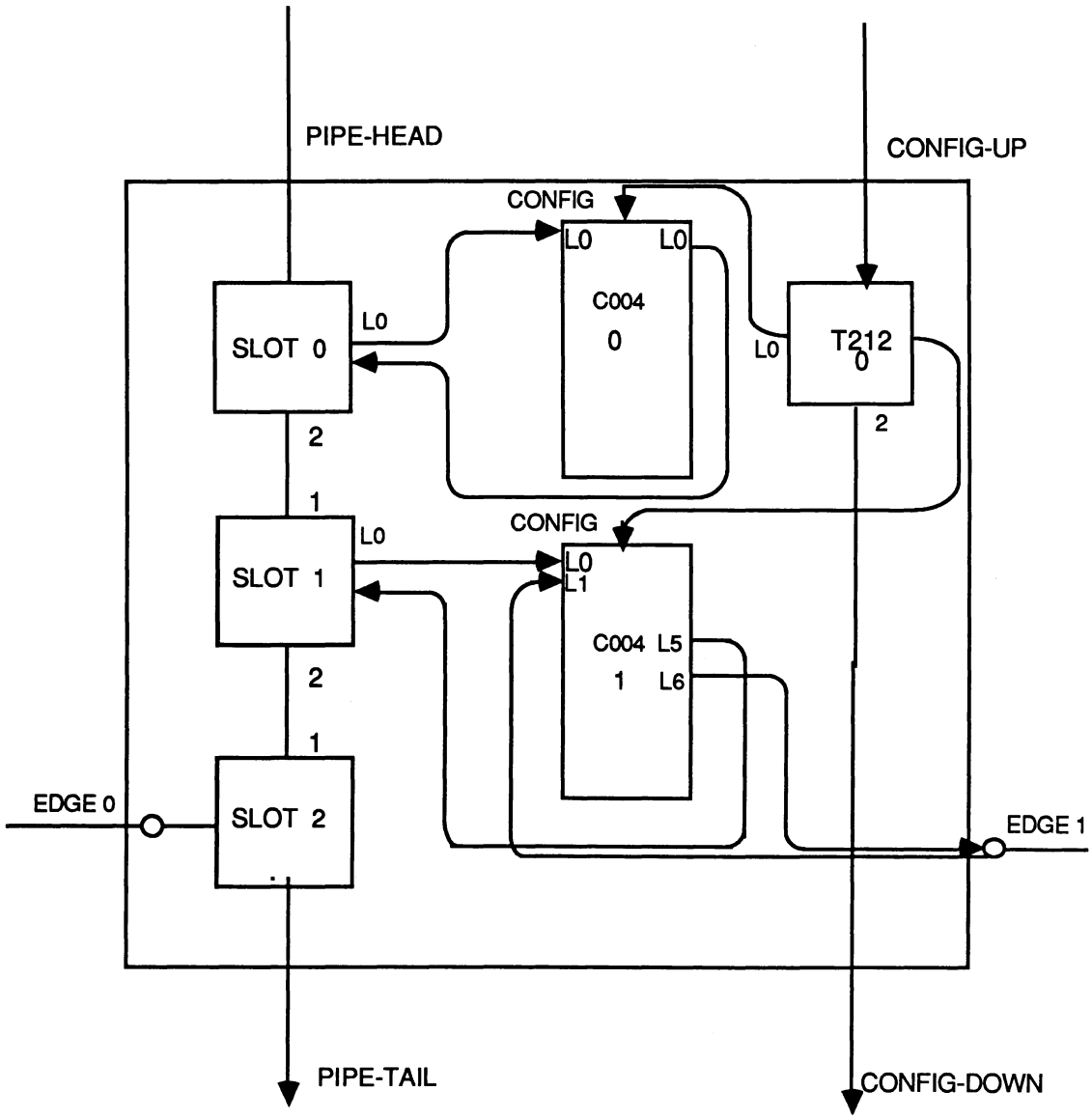
```
DEF myboard

SIZES
  T2   1
  C4   2
  SLOT  3
  EDGE  2
END

T2CHAIN
  T2 0, LINK 0 C4 0
  T2 0, LINK 3 C4 1
END
```

```
HARDWIRE
  SLOT 0, LINK 2 TO SLOT 1, LINK 1
  SLOT 1, LINK 2 TO SLOT 2, LINK 1
  SLOT 1, LINK 3 TO EDGE 0
  C4 0, LINK 0 TO SLOT 0, LINK 0
  C4 1, LINK 0, I TO SLOT 1, LINK 0, O
  C4 1, LINK 1, I TO EDGE 1, O
  C4 1, LINK 5, O TO SLOT 1, LINK 0, I
  C4 1, LINK 6, O TO EDGE 1, I
END

PIPE myboard END
```



It can be seen in this example that a hardwire fold consists of two main components : a section where board types are defined, and a section where the combination of these board types in a pipeline is described.

The pipeline description section is a simple list of the motherboards that are linked together to form a system, in the case of the previous example just a single board of type **myboard**. Boards are mentioned in the **PIPE** section in the same order as they appear in the pipeline. For example:

```
... DEF b00x
... DEF myboard

PIPE myboard, b00x, b00x END
```

describes a pipeline of three motherboards, headed by a board of type **myboard** and followed by two **b00xs** (for an introduction to the way that INMOS motherboard architecture permits this, read the 'Motherboard Architecture Manual.'

The board definition section may contain a number of board type definitions, each of which has four parts: a board type name, a sizes section, a t2chain section and a hardwire section.

Each board definition starts with the reserved word **DEF** followed by a user-defined board type name.

The 'sizes' section, which must follow the order of the example, is a simple list of the numbers of T2s, C004s etc in a system.

The second, 'T2chain' section describes the way that C4 ids (which are used later on) relate to the links coming off the T212/C004 config pipeline.

The hardwires section describes the link connections between slots, edge links and C004s.

It may contain any number of four different types of statement, in any order. These statement types describe slot to slot connections, slot to edge connections, C004 to slot connections and C004 to edge connections.

All follow the same simple format. For example:

```
SLOT 7, LINK 0 TO SLOT 8, LINK 3
```

describes a hardwired connection between link 0 on slot number 7 and slot number 8, link 3.

```
C4 1, LINK 3 TO SLOT 4, LINK 3
```

describes a connection between link 3 on C004 number 1 and link 3 on slot 4.

```
        {}
C4 2, LINK 28 TO EDGE 7
```

describes a link connection between link 28 on C004 number 2 and EDGE link 7.

```
SLOT 1, LINK 3 TO EDGE 2
```

describes a link connection between slot 1, link 3 and EDGE link 2.

Slot-to-slot and slot-to-edge connections are made as links - both inputs and outputs are assumed to be connected as one unit.

Connections to C004s can also be described in the same way. However, if necessary, these connections can split into two separate 'hard channels', allowing any single link to be routed via two C004s. This is done using the reserved words **I** and **O**. For example :

```
C4 1, LINK 22, O TO SLOT 3, LINK 0, I
```

states that the output of link 22 on C004 number 1 is connected to the input of link 0 on slot 3.

Note that all uses of the word **LINK** in hardwire and softwires folds are optional.

4 The Softwire Fold

4.1 Introduction

A valid softwires fold consists of **SOFTWIRE** followed by any number of *board.softwire.sections* followed by **END**.

A *board.softwire.section* consists of **PIPE** followed by a *board.i.d.*, followed by any number of *softwire.lines*. The *board.id* is a integer which represents the board to which the *softwire.lines* refer. This would be '0' for the first board in the pipe, '1' for the second etc.

A *softwire.line* can specify a slot to slot connection, a slot to edge connection, or an edge to edge connection.

A *slot.to.edge.line* looks very like the equivalent construct in the hardwire fold.

The first section of a softwire *slot.to.slot.line* resembles the equivalent in the hardwire fold. However it differs in that it has another section added to the end, the *via.edges.section*.

A *via.edges.section* can be either *empty* (non existent), or can specify two edge ids. The first of these edge links will be soft linked to the leftmost slot mentioned in the line, the second to the second in the line.

The purpose of including this is to allow the user easily to set up soft connections indirectly via edge links, where board architecture does not permit direct connection.

The effect of adding a *via.edges.section* to the end of a *slot.to.slot.line* is exactly the same as preceding the slot to slot line with two *slot.to.edge* lines which specify the slot to edge connections explicitly. For example :

```
SOFTWIRE
  PIPE 0
     .
     .
     SLOT 2, LINK 0 TO SLOT 1 LINK 0 VIA EDGE 1,2
     .
     .
END
```

would have exactly the same effect as :

```
SOFTWIRE
  PIPE 0
     .
     .
     SLOT 2, LINK 0 TO EDGE 1
     SLOT 1, LINK 0 TO EDGE 2
     EDGE 1 TO EDGE 2
     .
     .
END
```

An *edge.to.edge.line* simply specifies two edges which are to be connected, for example:

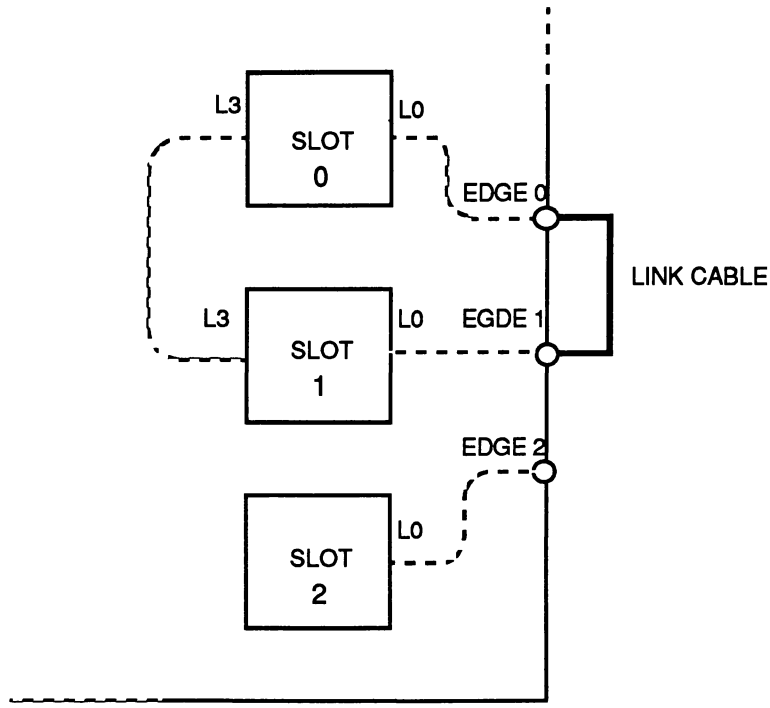
```
EDGE 4 TO EDGE 7
```

specifies that edge 4 is to be connected to edge 7.

4.2 An Example Softwires Fold

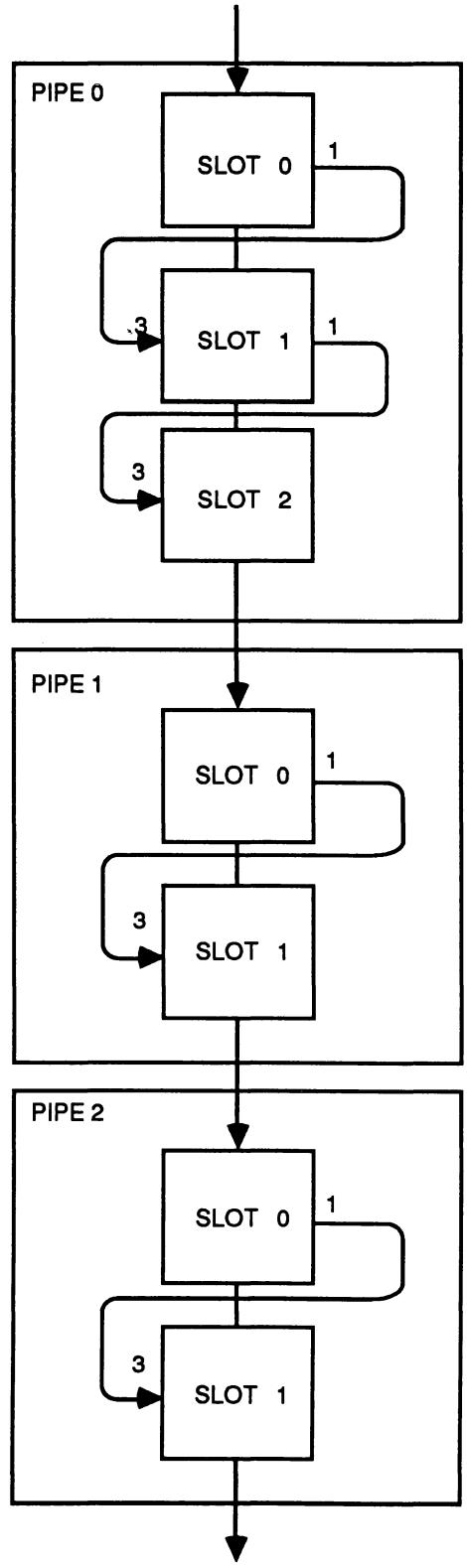
To provide an example of a complete fold using all of these constructs, the following softwires fold specifies all the connections in the diagram below :

```
SOFTWIRE
  PIPE 0
  SLOT 0, LINK 3 TO SLOT 1, LINK 3
  SLOT 0, LINK 0 TO SLOT 1, LINK 0 VIA EDGE 0,1
  SLOT 2, LINK 0 TO EDGE 2
END
```



The following softwires fold specifies connections in a three board system:

```
SOFTWIRE
  PIPE 0
    SLOT 0, LINK 0 TO SLOT 1, LINK 3
    SLOT 1, LINK 0 TO SLOT 2, LINK 3
  PIPE 1
    SLOT 0, LINK 0 TO SLOT 1, LINK 3
    SLOT 1, LINK 0 TO SLOT 2, LINK 3
  PIPE 2
    SLOT 0, LINK 0 TO SLOT 1, LINK 3
    SLOT 1, LINK 0 TO SLOT 2, LINK 3
```


PIPE 0
SLOT 0
1
SLOT 1
1
3
SLOT 2
3
PIPE 1
SLOT 0
1
SLOT 1
3
PIPE 2
SLOT 0
1
SLOT 1
3

5 Error Reporting

5.1 Errors Detected In The Hardwire Fold

There are a number of different types of error that may be detected by the MMS when reading the hardwire fold:

- File reading errors
- Syntax errors
- Range checking errors
- Duplication errors

In most cases any error messages produced should be self explanatory.

File Reading Errors

If the MMS is run when not pointing at a fold bundle containing at least one filed source text fold an error will be reported and explained. In some circumstances errors of this type will be detected first as a syntax error and reported as such.

Syntax Errors

Any syntax errors in the hardwire fold (as defined in the previous section) will be reported, producing one of three syntax error message types :

'... unexpected symbol found...' - meaning that some recognised keyword was found, but a different one was expected.

'... unexpected number found...'

'... unrecognised word found...'

The word that was expected at that point is usually displayed as well, together with the source line number that the mistake was found on. This line is also displayed in full below the error message. By using the **V** (view) command the user can then, if necessary, view the line together with the 4 lines which surround it.

For example, if the 'sizes' section of the hardwire fold looked like this :

```
SIZES
  T2  2
  C4  4
  SLOT  32
END           -- should be EDGES line !
```

The MMS would produce the following error message :

```
Error detected in HL1 fold at line 4 :
- Unexpected symbol found ('END'). 'EDGE' was expected

Line 4 : END          -- should be EDGES line  !
```

Commanding 'view' for line 4 would display the following :

```
2 :   C4  2
3 :   SLOT 32
4 : END         -- should be EDGES line !
5 :
6 :
```

Range Checking Errors

If a number is used which is outside the implementation limits defined in the help screen for the help command, or which is outside the limits defined in the 'SIZES' section, or which describes a link number greater 3 or less than 0 an 'integer out of range' will be reported. Again, the source line on which it was detected is also displayed.

Duplication Errors

If any link on a slot, C004 or edge is mentioned more than once in the 'hardwires section' an error message will be generated which gives details about this of duplication. Double mentions of T212 links or C004 IDs. in the **T2CHAIN** section are dealt with similarly.

For example, if the 'hardwire' section contains the following lines :

```
.
.
C4 0,LINK 4,O TO SLOT 4, LINK 3,I
.
.
C4 0,LINK 4,O TO SLOT 7, LINK 0,I
.
.
```

Then the MMS would produce the following error messages :

```
Error detected in HL1 fold at line x :
- The C004 link in this connection is already involved
in a C004 to slot connection

Line x : C4 0,LINK 4,O TO SLOT 7, LINK 0,I
```

It is worth being aware that the links involved in a connection are not necessarily checked for duplication in the same order that they appear in the line.

5.2 Errors Detected In The Softwire Fold

Syntax, file reading, range checking and duplication errors are handled in a very similar way to that used for the hardwire fold.

In addition to duplication, range and syntax errors however, the MMS will also report soft connections which it is unable to establish.

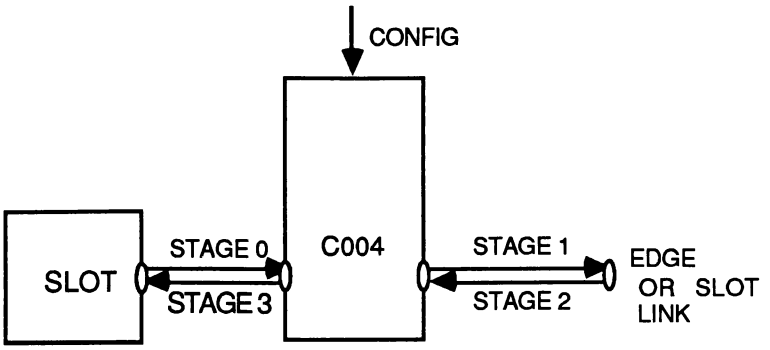
This can be for one of two reasons : firstly that a 'hard link' which is mentioned in a soft connection statement is not defined as connected anywhere in the hardwire fold : for example, if a

```
'SLOT 1,LINK 0 TO EDGE 3'
```

soft connection was specified, but slot 1, link 0 had not been mentioned anywhere in the hardwire fold.

The second possible cause is a 'C004 mismatch'. This occurs if two hard channels are required to be softwired together but are defined as connected to different C004s (making their soft connection together impossible).

To make it easier to report and correct such errors the MMS error messages break the process of establishing a soft link down into four stages. An error can be detected and reported as occuring at any of these stages. They are :



Stage 0 : "From 'from link' output to C004 input."

Stage 1 : "From C004 output to 'to link' input."

Stage 2 : "From 'to link' output to C004 input.

Stage 3 : "From C004 output to 'from link' input.

For example in the softwire line :

```
'SLOT 0,LINK 3 TO SLOT 1,LINK 0'
```

stage 0 is the connection from slot 0, link 3 output to a C004 input, stage 1 is the connection from the C004 output to slot 1,link 0 input. Stage 2 is from slot 1, link 0 output to C004 input, stage 3 is C004 output to slot 0 link 3 input.

If, in the above example, slot 0, link 3 output was not defined as being connected anywhere at all, the following error message would be produced :

```
Error while generating connection for line 1 :
A C004 connection could not be established at stage 0
i.e. from 'from-link' output to C004 input
reason : This hard channel is not connected to a C004

Line x : SLOT 0,LINK 3 TO SLOT 1,LINK 0
```

The four stages are applied in the same way to slot to slot or slot to edge connections.

A Hardwire Description

A.1 Notation

A BNF style syntax definition is used here to define the syntax of hardwire.

The following symbols are meta-symbols belonging to the BNF formalism, not to HL1 :

= means "is defined as"
| means "or"
{ } enclose items which may be repeated zero or more times
[ ] enclose items which are optional

All other symbols are part of the language. Each syntactic construct is printed in *italics* e.g.*sizes.section* and *hardwires.line*. Reserved words are printed in **boldface** e.g. **EDGE** and **SLOT**.

N.B. This implementation makes all uses of the reserved word **LINK** optional, in both the hardwire and the softwire fold.

For example the line :

```
SLOT 2, LINK 0 TO SLOT 0, LINK 3
```

could be written as :

```
SLOT 2,0 TO SLOT 0,3
```

A.2 BNF Syntax Definition

valid.hl1.fold = *board.definition.section pipeline.description*

pipeline.description = **PIPE** { *user.defined.board.name* }

board.definition.section = { *board.definition* }

board.definition = **DEF** *user.defined.board.name sizes.section t2chain.section hardwires.section*

sizes.section = **SIZES**
T2 *integer*
C4 *integer*
SLOT *integer*
EDGE *integer*
END

integer is a positive integer within limits varying between implementations.

t2chain.section = **T2CHAIN** { *t2/c4.line* } **END**

t2/c4.line = **T2** *t2.id*, [**LINK**] *link.number* **C4** *c4.id*

t2.id is an integer id associated with one T212 in the system

link.number is an integer between 0 and 3 identifying a link on the T212

c4.id is an integer id associated with one C004 in the system

hardwires.section = **HARDWIRE** { *hardwires.line* } **END**

hardwires.line = *slot.to.slot.line* | *c4.to.slot.line* | *c4.to.edge.line* | *slot.to.edge.line*

slot.to.slot.line = **SLOT** *slot.id*, **[LINK]** *link.number***TO SLOT** *slot.id* , **[LINK]** *link.number*

slot.id is a positive integer id associated with one slot in the system.

link.number is an integer id from 0 to 3 which identifies one link.

c4.to.slot.line = *simple.c4.to.slot.line* | *i.and.o.c4.to.slot.line*

simple.c4.to.slot.line = **C4** , *c4.id*, **[LINK]** *c4.link.no*, *I/O* **TO SLOT** *slot.no* **[LINK]** *link.number*, *I/O*

i.and.o.c4.to.slot.line = **C4** , *c4.id*, **[LINK]** *c4.link.no* **TO SLOT** *slot.no* **[LINK]** *link.number*

I/O = **I** | **O**

c4.link.number is a non-negative integer identifying a link on a C004.

c4.to.edge.line = *simple.c4.to.edge.line* | *i.and.o.c4.to.edge.line*

simple.c4.to.edge.line = **C4** *c4.id*, **[LINK]** *c4.link.number*, *I/O* **TO EDGE** *edge.id*, *I/O*

i.and.o.c4.to.edge.line = **C4** *c4.id*, **[LINK]** *c4.link.number*, **TO EDGE** *edge.id*

edge.id is an integer id associated with one edge link in the system

slot.to.edge.line = **SLOT** *slot.id*, **[LINK]** *link.number* **TO EDGE** *edge.id*

A.3 Description Of Use

A valid hardwire fold consists of two main parts, a set of board definitions and a pipeline description. The pipeline description is simple list of the board types which are linked together to form a motherboard system, mentioned in order of appearance.

The first *'sizes.section'* states the number of slots, C004s T212s and edges used in the system.

The second, *'t2chain.section'* describes the way that the C004 link switches are connected to the spurs of the T212/C004 pipeline.

The third, *'hardwires.section'*, defines the link connections between C004s, slots and edges.

The *sizes.section* is a straightforward list of the numbers of T212s, C004s, slots and EDGES in the system. For example :

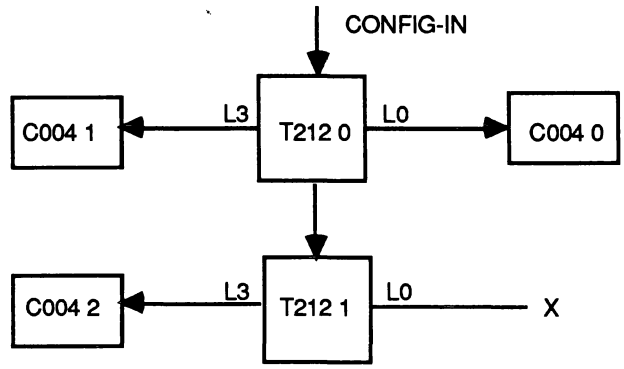
```
SIZES
  T2   3
  C4   6
  SLOT  12
  EDGE  6
END
```

describes a system composed of 3 T212s controlling 6 C004s, with 12 slots and 6 edge connected links.

The *t2chain.section* consists of '**T2CHAIN**' followed by any number of *t2/c4.line*s followed by '**END**'. For example :

```
T2CHAIN
  T2 0,LINK 0  C4 0
  T2 0,LINK 3  C4 1
  T2 1,LINK 0  C4 2
END
```

describes a system whose config pipeline looks like this:

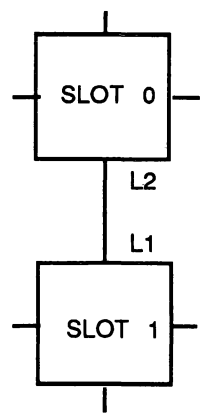


The hardwired link connections in a system are described in the *hardwires.section* by a series of *hardwires.line*s. Each of these lines can be describe a connection between slots, between a slot and a C004, between an edge link and a C004 or between a slot and an edge link.

A *slot.to.slot.line* describes a hardwired connection between a link on one slot and a link on another. For example :

```
HARDWIRE
  .
  .
  SLOT 0,LINK 2 TO SLOT 1,LINK 1
  .
  .
END
```

describes this connection :



A *c4.to.slot.line* describes a connection between a C004 link and a slot.

Such a connection can be described in two different ways :

- *simple.c4.to.slot.line*
- *i.and.o.c4.to.slot.line*

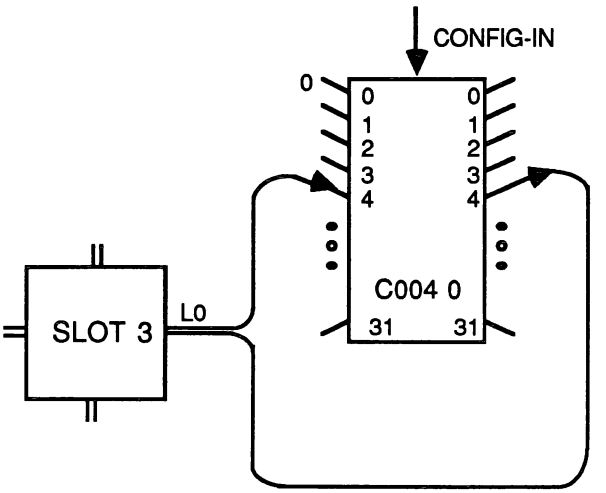
The former, like the *slot.to.slot.line*, describes the whole link connection as one unit. The latter however defines each link as two 'hard channels'. For example :

```
HARDWIRE
  .
  .
  C4 0, LINK 4,I TO SLOT 3, LINK 0,O
  C4 0, LINK 4,O TO SLOT 3, LINK 0,I
  .
  .
END
```

which has the same meaning as :

```
HARDWIRE
  .
  .
  C4 0, LINK 4 TO SLOT 3, LINK 0
  .
  .
END
```

describes these connections :



A *c4.to.edge.line* is similar to a *c4.to.slot.line* except that it defines a connection to an edge connected link rather than a link on a slot. Again the connection can be described either as two 'hard channels' or as a single link. For example :

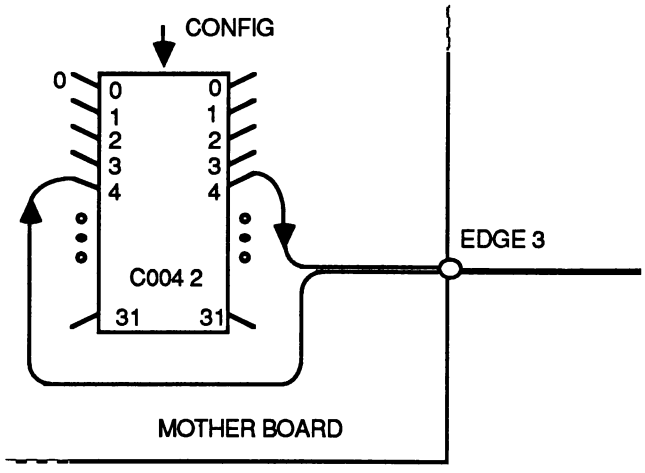
```
HARDWIRE
  .
  .
```

```
    C4 2, LINK 4,I TO EDGE 3,O
    C4 2, LINK 4,O TO EDGE 3,I
    .
    .
END
```

which has the same meaning as :

```
HARDWIRE
    .
    .
    C4 2, LINK 4 TO EDGE 3
    .
    .
END
```

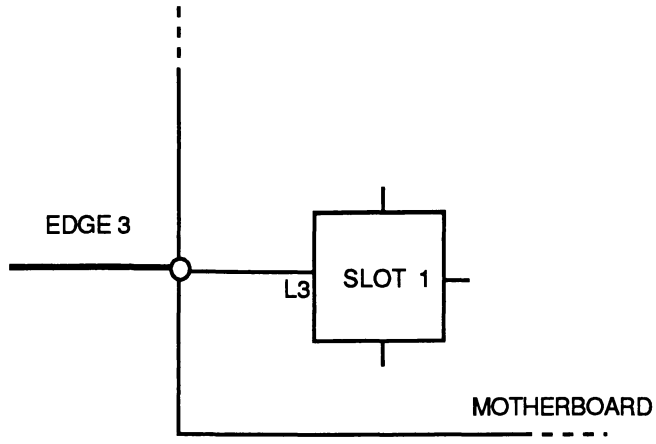
defines these connections :



A *slot.to.edge.line* is similar to a *slot.to.slot.line* except that it defines a link connection to an edge link. For example :

```
HARDWIRE
    .
    .
    SLOT 1, LINK 3 TO EDGE 3
    .
    .
END
```

defines this connection :



B Softwires Fold BNF Syntax Definition

The following is a full BNF style definition of the syntax of a softwires fold.

valid.softwires.fold = **SOFTWIRE**
board.softwires.section
END

board.softwires.section = **PIPE** *board.i.d.* { *softwires.lines* }

board.id is a non.negative integer between limits varying between implementations.

softwires.line = slot.to.slot.line |slot.to.edge.line |edge.to.edge.line

slot.to.edge.line = **SLOT** *slot.id*, **[LINK]** *link.number* **TO EDGE** *edge.id*

slot.to.slot.line = slot.to slot.section via.edges.section

slot.to.slot.section = **SLOT** *slot.id*, **[LINK]** *link.number* **TO SLOT** *slot.id* , **[LINK]** *link.number*

via.edges.section = empty | **VIA EDGE** *edge.id*, *edge.id*

edge.to.edge.line = **EDGE** *edge.id* **TO EDGE** *edge.id*

C The C004-A Programmable Link Switch

The IMS C004-A is a 'link switching' chip, produced by INMOS as part of the transputer family. Its function is to allow the switching of link connections between transputers, under software control.

Each IMS C004-A is able to switch 32 links as an unrestricted 'crossbar'. This means that, using a single IMS C004-A chip, any one of 32 transputer links may be connected to any other. A single IMS C004-A, for example, could allow the 'soft' configuration of any possible network of 8 transputers.

The switching of links on an IMS C004-A is controlled by a standard transputer link known as the 'config' link. This link will respond to a number of commands :

0 Connect channel. The sequence '1 ; x ; y' will connect input 'hard channel' x to output 'hard channel' y.

1 Connect link. The sequence '0 ; x ; y' will connect link x to link y. This operation is thus equivalent to connecting input x to output y and input y to output x.

2 Enquire. Sending the sequence '2 ; x' would cause the C004-A to respond by sending back a byte representing the input that ouput x is connected to.

3 Nop. Sending a '3' performs no operation, but is useful for synchronisation. After receipt of a configuration sequence, a connection will not become reliably established immediately. Sending a '3' provides the necessary delay.

4 Resets the IMS C004-A, disconnecting and holding low all outputs.

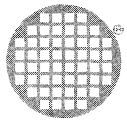
All commands and parameters are single byte. For example the following occam code fragment :

```
.
.
to.C004-A.config ! 1(BYTE)
to.C004-A.config ! 3(BYTE)
to.C004-A.config ! 7(BYTE)
.
.
```

would cause C004-A link 3 to be connected to C004-A link 7.

Note that applying reset to the IMS C004-A does not bring the link connections to any consistent state. To disconnect all outputs a software reset must be used.

Technical note 19, available from INMOS, describes the C004 and its application in more detail.



INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
UK
Telephone (0454) 616616
Telex 444723

INMOS Corporation
PO Box 16000
Colorado Springs
CO 80935
USA
Telephone (303) 630 4000
TWX 910 920 4904

INMOS GmbH
Danziger Strasse 2
8057 Eching
Munich
West Germany
Telephone (089) 319 10 28
Telex 522645

INMOS Japan K.K.
4th Floor No 1 Kowa Bldg
11 - 41 Akasaka 1-chome
Minato-ku
Tokyo 107
Japan
Telephone 03-505-2840
Telex J29507 TEI JN
Fax 03-505 2844

INMOS SARL
Immeuble Monaco
7 rue Le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 46.87.22.01
Telex 201222

72 TDS 143 00
January 1988