Header Sheet

This sheet is required for the configuration control of this document.

It is not a part of the delivered document.

Title: {Title, Summary Block }	Resource Management Configuration and Diagnostics	
Doc. No:	5000 xxxx.00	{Subject/Rev, Summary Block }
Date:	21 June, 1996	{Create Date, auto}
File:	RM_CNFG.DOC	{Filename, auto}
Directory:		(Server Directory, enter when filing)
Prepared by:	Matthew Gream	{Author, Summary Block }
		(signature)
Reviewed by:		(Reviewers Name, enter at start)
		(signature)
Approved by:		(Approvers Name, enter at start)
		(signature)

Document History:

(Enter when filing)

Rev.	ECN	Date	Name	Page(s)	Description
01					
02					

ENGINEERING DESCRIPTION

Resource Management Configuration and Diagnostics

June, 96

No. 5000 xxxx.00
A.C.N. 003 169 088

Table of Contents

1. Introduction

The purpose of this document is to describe the configuration and diagnostic mechanisms available for the RM's Resource Management software.

The description provides a brief overview of Resource Management, then outlines the configuration and diagnostics mechanisms that are available. After this, it describes common processes and provides an application example.

The intended audience for this document is engineering.

2. Overview

This section provides a brief overview of where Resource Management is placed in the system, and what its role is.

2.1. Software Architecture

Resource Management exists as a lower layer in the Software Architecture. Its purpose is to provide abstract Services and Connections to higher layer application entities by using and assembling primitive distributed Resources. These Resources make themselves available to be used by Resource Management. They may exist on various physical platforms.

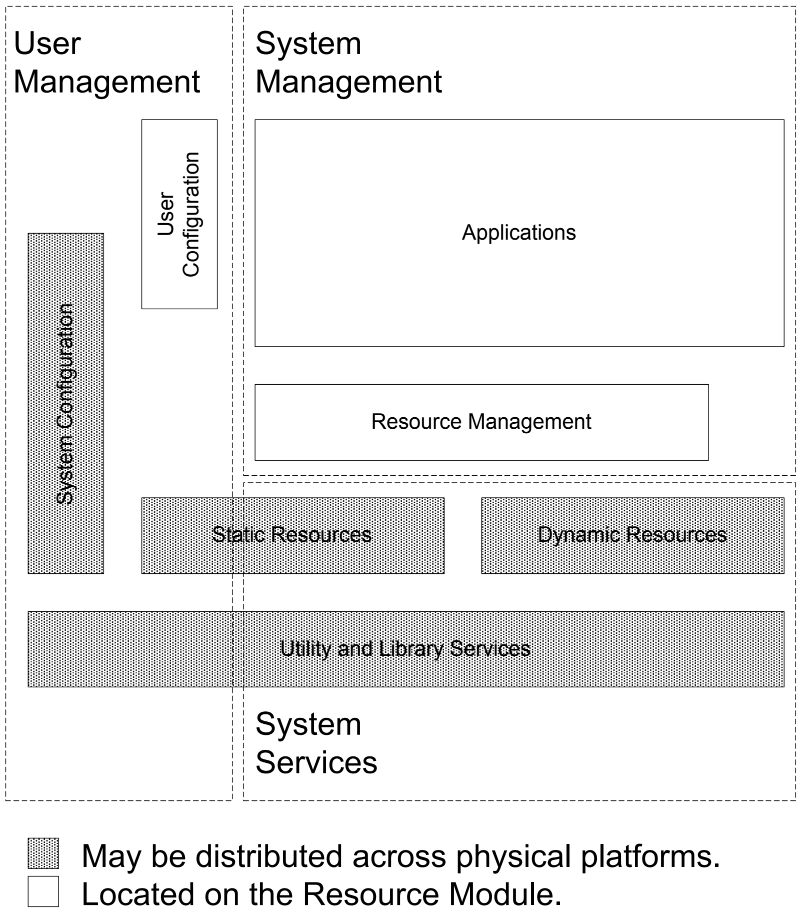

Figure 1. Software Architecture

The distributed Resources conform to a simple protocol that allows them to be interconnected and associated with each other across physical boundaries. Resource Management shields applications from distributed resource control and interconnection issues (i.e. interprocessor boundaries, interprocessor data transfer, bandwidth allocation, physical communication failures, etc). Services and Connections are used by applications via. a simple protocol.

2.2. Resource Management Architecture

Resource Management consists of several key entities as illustrated in the following diagram.

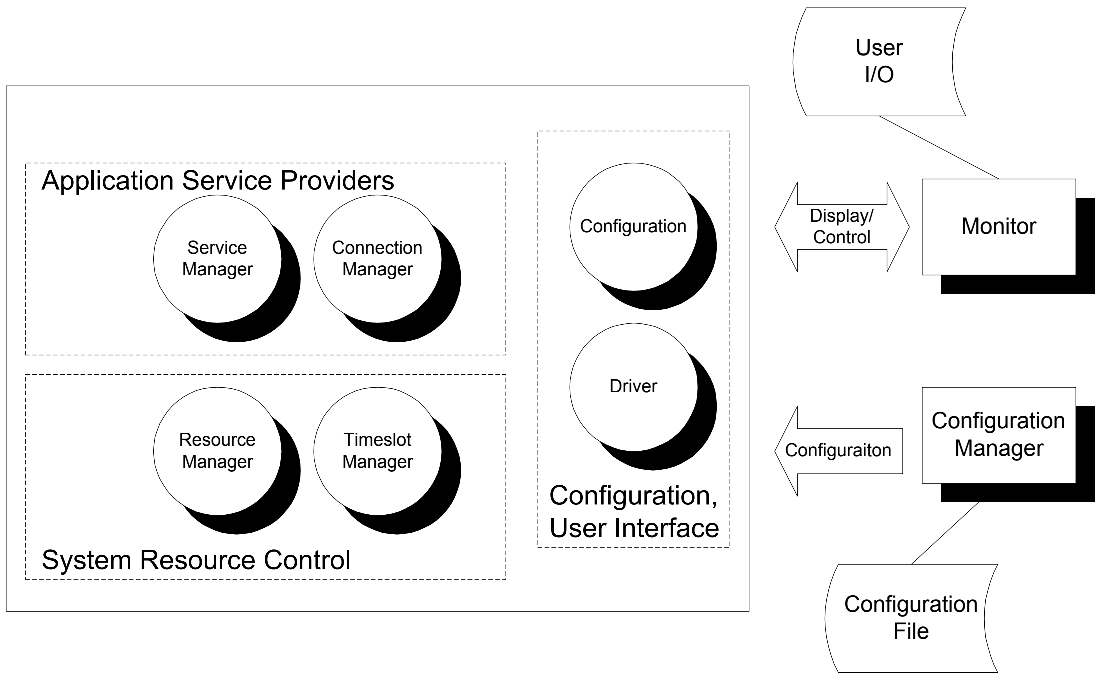

Figure 2. Resource Management Architecture

1. Resource Manager -- controls the distributed Resources by allowing the Service Manager to allocate, manipulate and deallocate the Resources. Resources may require Timeslots from the Timeslot Manager.

2. Service Manager -- provides application level services by utilising Resources and Connections. This produces a functional entity that makes sense to an application (e.g. protocol stack).

3. Connection Manager -- implements application level connections, and lower level connections by using the Timeslot Manager, and interacting with Service and Resource Managers.

4. Timeslot Manager -- allocates system wide Timeslots to Resources.

5. Configuration Task -- initiates start up Services and provides a monitor interface for activating and deactivating particular Services.

6. Driver Task -- provides user accessible driver functions to manipulate and interact with the Resource Management software.

7. Configuration File -- contains configuration used by the Resource Management software as provided by the facilities in the Configuration Manager.

8. User I/O -- illustrates the placement of user input and output via. the external Monitor task.

2.3. Resource Management Operation

Configuration

When Resource Management commences operation, it reads information from a configuration file. This information is used to define characteristics of Services and Resources present in, and controlled by, Resource Management.

Resources

Resources may Sign On and Sign Off at any point in time, reflecting their availability in the system. When used by Resource Management, they are configured, activated and associated with each other. This may occur irrespective of the physical platform upon which they reside.

Services

Requests can be made to activate Services, which will succeed if the Service is defined, and all Resources required for the Service are available. The Service definition comes from either a configuration file, or as implemented by hard coded classes. A Service will generally use a number of Resources and Connections in its operation.

Connections

Users can institute Connections between Services, they do this by requesting that Connections are made between interfaces on active Services.

3. Configuration

Configuration is required in order to specialise the behaviour of Resource Management. This occurs in two ways: Hard Configuration, and Soft Configuration. The former refers to modifications and specialisations of the software, whereas the latter refers to a flexible configuration file.

3.1. Hard Configuration

This form of configuration is chosen when behaviours are required that cannot be accommodated by the Soft Configuration. Modifications can be made to particular classes. The configuration is implemented by building the changes into the software.

3.1.1. Resources

Resources represent the interface to the distributed entities that are provided to the Resource Manager and used by Services in the Service Manager.

This will be completed when required.

3.1.2. Services

Services represent aggregated functional components that provided by the Service Manager. These are used the application or Connections.

This will be completed when required.

3.1.3. Connections

Connections represent the high level associations that bind together Services, or Resources. These are used by the application or Services.

This will be completed when required.

3.1.4. Nodes

Nodes are the Connection Manager's representation of Resources to which Links are bound. These are used by Links.

This will be completed when required.

3.1.5. Links

Links are the Connection Manager's representation of a single connection between two nodes. These are used by Connections.

This will be completed when required.

3.1.6. Interfaces

Interfaces are the Connection Manager's representation of a connectable end point. These are used by Resources and Services in interaction with the Connection Manager.

This will be completed when required.

3.2. Soft Configuration

This form of configuration is chosen when the required behaviours can be accommodated within the limitations of the soft configuration (i.e. a simple set of rules for describing how the objects should operate).

Modifications are made to a configuration file that is read when the software is started. It cannot be re-read during the life of the software, therefore the software must be restarted to implement any changes.

There are essentially four areas of configuration: Resources, Services, Connections and Configuration.

3.2.1. Resources

Resources can have characteristics described in the configuration file, these are read and stored in Resource Configuration Objects, and utilised by a Generic Resource class.

3.2.1.1. Description

The configuration file contains an entry for each Resource. The entry describes the Resource, and specifies Interfaces or Modes for that Resource. An Interface must be present for Connections to be made to the Resource (i.e. it describes an Upper, Lower or other interface). Modes are used as a way of encapsulating a certain configuration for the Resource (i.e. operating a Transport Protocol in Reliable or Unreliable Mode). There can be any number of Interfaces and Modes defined for a Resource.

Modes and Connections become important when the Resource is used as part of a Service. The Service can allocate the Resource in a particular Mode and make Connections between the Interfaces advertised by the Resource.

3.2.1.2. Grammar

The following grammar is used to specify a Resource.

```
resource_definition :=
    resource_entry_header { resource_entry }

resource_entry :=
    resource_entry_name |
    resource_entry_copy |
    resource_entry_mode |
    resource_entry_version |
    resource_entry_interface |
    resource_entry_interface_config;

resource_entry_header :=
    '[' 'RM-Resource' '-' resource_id '-' resource_copy '-' module_id ']';

resource_entry_name :=
    'Name' '=' resource_name;

resource_entry_copy :=
    'Copy' '=' resource_copy;

resource_entry_mode :=
    'Mode' '=' resource_mode ',' { byte ',' };

resource_entry_version :=
    'Vers' '=' resource_version;

resource_entry_interface :=
    resource_entry_software_interface |
    resource_entry_timeslot_interface;

resource_entry_software_interface :=
    'IIns' '=' interface_id ',' 'S' ','
               interface_id ',' bandwidth ',' bandwidth;

resource_entry_timeslot_interface :=
    'IIns' '=' interface_id ',' 'T' ','
               interface_id ',' count ','
                        { timeslot_channel ',' timeslot_mask ',' };

resource_entry_interface_config :=
    'IIni' '=' interface_id ',' { byte ',' };


resource_id := HEX(0x0000..0xFFFF);
resource_name := STRING(0..16);
resource_copy := HEX(0x00..0xFF);
```

```
resource_mode := HEX(0x00..0xFF);
resource_version := HEX(0x000..0xFFF);
byte := HEX(0x00..0xFF);
interface_id := HEX(0x00..0xFF);
bandwidth := HEX(0x00000000..0xFFFFFFFF);
timeslot_channel := HEX(0x0000..0xFFFF);
timeslot_mask := HEX(0x00..0xFF);
module_id := HEX(0x00..0xFF);
count := HEX(0x00..0xFF);
```

3.2.1.3. Notes

1. The two bandwidth items in the Software Interface are used to specify a minimum and maximum bandwidth that is acceptable for connections to the interface.

2. When specifying a Timeslot Interface, only the count need be specified which will cause a linear set of timeslots to be allocated.

3. The Module Id in the Header is not used at this point in time.

4. If the Version is specified, then a Sign On with a lower version number will be rejected.

3.2.1.4. Examples

The following examples illustrate actual Resources from a working configuration file.

1. DPFPGA Resource -- Software and Timeslot Interfaces

This Resource is given a pseudo identifier of 0xF003, it corresponds to the DPFPGA Hardware. To connect to other Resources, it provides two interfaces: an Upper Layer Software Interface and a Lower Layer Timeslot Interface. The Lower Layer operates at 2Mbps. When the Resource performs a Sign On, the Resource Manager will attempt to provide 2Mbps worth of Timeslots for it.

```
; =====================================================================
;   DPFPGA
;   RM_RESOURCE_ID_HW_DPFPGA                rscId (0xF003)
;    Modes
;       -
;    Interfaces
;       - U(S) -- 2Mbps (any at the moment)
;       - L(T) -- 2Mbps (32 timeslots)
;    Notes
;       -
; =====================================================================
[RM-Resource-F003-00-RM]
Name=Hw DPFPGA
IIns=U,S,U,0,FFFFFFFF
IIns=L,T,L,20
```

2. Datalink Transparent Stack Resource -- Software and Timeslot Interfaces

The Resource, also a pseudo Resource having an identifier 0xE008, has no Interfaces (because it is fixed to communicate with other entities in the Datalink). It does however supports four Modes. Each Mode contains two PIDs which configure the Rx and Tx Buffer Sizes. This allows a Service to request the Resource in one of these four Modes, and know that the appropriate Buffer Sizes will be set, without the Service having to be aware of the PID values.

```
; ======================================================================
;   Datalink - Transparent Stack Layer
;   RM_RESOURCE_ID_DL_STK_TRANSPARENT       rscId (0xE008)
;    Modes
;       - 00: 10 byte MTU/MRU
;       - 01: 248 byte MTU/MRU
;       - 02: 1514 byte MTU/MRU
;       - 03: 4096 byte MTU/MRU
;    Interfaces
;       -
;    Notes
;       -
; ======================================================================
[RM-Resource-E008-00-RM]
Name=DL Stk Trans
Mode=00,40,04,00,00,00,0A,42,04,00,00,00,0A
Mode=01,40,04,00,00,00,F8,42,04,00,00,00,F8
Mode=02,40,04,00,00,05,FA,42,04,00,00,05,FA
Mode=03,40,04,00,00,10,00,42,04,00,00,10,00
```

3.2.2. Services

Services can have characteristics described in the configuration file, these are read and stored in Configuration Objects, and utilised by a Generic Service class.

3.2.2.1. Description

The configuration file contains an entry for each Service. The entry describes the Service, and specifies all of its Resource, Connection and Interface items. A Resource item specifies a Resource, its mode, and additional configuration to be passed to the Resource when it is Reserved. A Connection specifies which two Resources are to be connected, and may also provide additional configuration information. A Service Interface item translates to an Interface on a specified Resource.

The Interfaces become important when users request Connections between Service Instances. The Interfaces are mapped to Resource Interfaces, and then the Connections are made. This allows for applications to make Connections at an abstract level and then have them implemented at a lower level.

3.2.2.2. Grammar

The following grammar is used to specify a Service.

```
service_definition :=
    service_entry_header { service_entry }

service_entry :=
    service_entry_name |
    service_entry_copy |
    service_entry_resource |
    service_entry_resource_config |
    service_entry_connection |
    service_entry_connection_config |
    service_entry_interface;

service_entry_header :=
    '[' 'SM-Service' '-' service_id '-' service_copy ']';

service_entry_name :=
    'Name' '=' service_name;

service_entry_copy :=
    'Copy' '=' service_copy;

service_entry_resource :=
    'RIns' '=' resource_number ',' resource_id ','
               resource_copy ',' module_id ','
               resource_mode ',' resource_options;

service_entry_resource_config :=
    'RIni' '=' resource_number ',' { byte ',' };

service_entry_connection :=
    'CIns' '=' connection_number ','
               resource_number ',' interface_id ','
               resource_number ',' interface_id ','
               connection_options;

service_entry_connection_config :=
    'CIni' '=' connection_number ',' { byte ',' };

service_entry_interface :=
    'IIns' '=' interface_id ','
               resource_number ',' interface_id;

service_id := HEX(0x0000..0xFFFF);
service_name := STRING(0..32);
service_copy := HEX(0x00..0xFF);
resource_number := HEX (0x00..0xFF);
connection_number := HEX (0x00..0xFF);
resource_options := HEX(0x0000..0xFFFF);
connection_options := HEX(0x0000..0xFFFF);
```

3.2.2.3. Notes

1. The Service and Connection numbers must start from zero and be in order.

2. The connection options are not yet used.

3. The following service options are available: 0x01 indicates to use the same copy for the resource, as is used for the service.

4. A Resource Copy of 0xFF indicates that "any" copy from those currently free and available can be used.

3.2.2.4. Examples

The following examples illustrate actual Services from a working configuration file.

1. Monitor on Front Panel Port 1 Service -- Datalink Service

This Service, allocated an Identifier of 0x0030, shows how several Resources are specified in sequential order, after which Connections are specified to bind the Resources together. Notice that the Datalink, in this case, uses a fixed Resource Copy of 2, however the Monitor can use any Copy.

```
; ======================================================================
;   Monitor on Front Panel Port 1
;   SM_SERVICE_ID_FPPORT1_MONITOR               srvId (0x0030)
;    Resources
;       - [0] : Sw Monitor
;       - [1] : DL Character Sap Layer
;       - [2] : DL Transparent Stack Layer (Mode 00: 10 byte)
;       - [3] : DL Uart Scc Mac Layer
;       - [4] : Hw Quicc Scc 3
;       - [5] : DL Task
;    Connections
;       - L(0) - U(1)
;       - L(3) - U(4)
;    Interfaces
;       -
;    Notes
;       -
; ======================================================================
[SM-Service-0030]
Name=FP Port 1 - Monitor
RIns=00,00EE,FF,00,00
RIns=01,E002,02,00,00
RIns=02,E008,02,00,00
RIns=03,E00A,02,00,00
RIns=04,F001,02,00,00
RIns=05,00F7,02,00,00
CIns=00,00,L,01,U,0000
CIns=01,03,L,04,U,0000
```

2. Transport Protocol on Loopback via. QMC Test Service -- Large Service

This Service, identifier 0x8201, shows how we can specify a large number of Resources, and have duplicate Resources as well. In this case, we are using fixed copies for the Datalink Resources. Some of the Resources are configured to operate in specific Modes.

```
; ======================================================================
;   Transport Protocol Loopback via. QMC test service -- RELIABLE
;       (Drop 5% of packets)
;                                               srvId (0x8201)
;    Resources
;       - [0] : Sw Terminal (Mode 02: Generator)
;       - [1] : Fp Transport (Mode 02: Reliable)
;       - [2] : DL JMessage Sap Layer
;       - [3] : DL Transparent Stack Layer (Mode 02: 1514 byte)
;       - [4] : DL QMC Mac Layer (Mode 02: HDLC)
;       - [5] : Hw Quicc Qmc 1
;       - [6] : DL Task
;       - [7] : Sw Terminal (Mode 01: Loopback)
;       - [8] : Fp Transport (Mode 02: Reliable)
;       - [9] : DL JMessage Sap Layer
;       - [10]: DL Transparent Stack Layer (Mode 02: 1514 byte)
;       - [11]: DL QMC Mac Layer (Mode 02: HDLC)
;       - [12]: Hw Quicc Qmc 2
;       - [13]: DL Task
;    Connections
;       - L(0) - U(1)
;       - L(1) - U(2)
;       - L(4) - U(5)
;       - L(7) - U(8)
;       - L(8) - U(9)
;       - L(11) - U(12)
;    Interfaces
;       -
;    Notes
;       -
; ======================================================================
[SM-Service-8201]
Name=TP (R) QMC Loopback
RIns=00,00ED,FF,00,02
RIns=01,00EC,FF,00,02
RIns=02,E005,11,00,00
RIns=03,E008,11,00,02
RIns=04,E00F,11,00,02
RIns=05,F005,11,00,00
RIns=06,00F7,11,00,00
RIns=07,00ED,FF,00,01
RIns=08,00EC,FF,00,02
RIns=09,E005,12,00,00
RIns=0A,E008,12,00,02
RIns=0B,E00F,12,00,02
RIns=0C,F005,12,00,00
RIns=0D,00F7,12,00,00
```

```
CIns=00,00,L,01,U,0000
CIns=01,01,L,02,U,0000
CIns=02,04,L,05,U,0000
CIns=03,07,L,08,U,0000
CIns=04,08,L,09,U,0000
CIns=05,0B,L,0C,U,0000
```

3. Fast Packet Encapsulation Generator on Front Panel Port 2 -- Extra Config Service

This Service illustrates how additional configuration PIDs can be specified and delivered to particular Resources in order to configure them. We are using the Datalink here, so the same copy is used for its Resources. Note that the PIDs are specified in a free form manner, and multiple lines can be used. In fact, arbitrary data can be placed instead of PIDs if required.

```
; ======================================================================
;   Fast Packet Encapsulation Generator on Front Panel Port 2
;                                              srvId (0x800A)
;    Resources
;       - [0] : Sw Terminal (w/ Generator Parameters)
;       - [1] : Fp Encapsulate
;       - [2] : DL JMessage Sap Layer
;       - [3] : DL Transparent Stack Layer (Mode 02: 1514 byte)
;       - [4] : DL Hdlc Fp Mac Layer
;       - [5] : Hw Quicc Scc 4
;       - [6] : DL Task
;    Connections
;       - L(0) - U(1)
;       - L(1) - U(2)
;       - L(4) - U(5)
;    Interfaces
;       -
;    Notes
;       - Generator:
;           reports: 5 sec , level 3, label "Encap",
;           pattern: fixed,
;               header: 7e -- don't put this
;               address: 00,00
;               ctrl,pad: 03,00
;               nlpid: 80
;               oui: 00,20,bc
;               pid: 00,01
;               data: 30, 31,32,33,34,35,36,37,38,39
;               fcs: 00,00 -- don't put this (but null terminate it)
;               trailer: 7e -- don't put this
;            (length: 22) -- 24 with header and trailer
;           packet:  size 24, time 1024 millisec.
;           upper: virtualcircuitid 00000001,00000002
;       - Encapsulation
;           Vcc: (pid: AB, 36 bytes)
;               id: 00,00,00,01
;               address: 00,00,00,02
;               cir: 00,00,00,00
;               numOfTxQ: 00,00,00,01
;               txQDataType: 00,00,00,01,
;                            00,00,00,00,
;                            00,00,00,00,
;                            00,00,00,00,
;                            00,00,00,00
; ================================================================
[SM-Service-800A]
Name=FP Port 2 - Fp Encap Generator
RIns=00,00ED,FF,00,00
RIni=00,A0,01,01
RIni=00,A1,01,03
RIni=00,A2,01,05
RIni=00,A3,06,45,6E,63,61,70,00
RIni=00,A4,15,00,00,03,00,80,00,20,BC,00,01
RIni=00,30,31,32,33,34,35,36,37,38,39,00
RIni=00,A7,02,00,18
RIni=00,A8,02,04,00
RIni=00,AA,08,00,00,00,01,00,00,00,02
RIni=00,AB,08,00,00,00,01,00,00,00,02
RIns=01,00AD,FF,00,00
RIni=01,AB,24,00,00,00,01,00,00,00,02,00,00,00,00
RIni=01,00,00,00,01,00,00,00,01,00,00,00,00,00,00
RIni=01,00,00,00,00,00,00,00,00,00,00,00
RIns=02,E005,03,00,00
RIns=03,E008,03,00,02
RIns=04,E00C,03,00,00
RIns=05,F001,03,00,00
RIns=06,00F7,03,00,00
CIns=00,00,L,01,U,0000
CIns=02,01,L,02,U,0000
CIns=01,04,L,05,U,0000
```

3.2.3. Configuration

The Configuration Task can read in configuration to specify which Services it should activate at startup, and those that can be activated via. the monitor. This means that when the task starts, it will attempt to activate the Services.

3.2.3.1. Description

There are three sets of configuration. The first set is used to specify all of the Service items that can be established by the Configuration Task. The second set is used to specify the lists of items that will be established when the Configuration Task commences operation. The third is used to specify which list will be the active one.

3.2.3.2. Grammar

The following grammar is used to specify the configuration items.

```
config_item := config_item_header { config_item_entry }

config_item_entry :=
    config_item_name |
    config_item_info |
    config_item_pids;

config_item_header :=
    '[' 'RM-Cfg-Item' '-' config_id ']';

config_item_name :=
    'Name' '=' config_name;

config_item_info :=
    'Info' '=' service_id ',' service_copy;

config_item_pids :=
    'Pids' '=' { byte ',' };

config_name := STRING(0..32);
config_id := HEX(0x00..0xFF);


config_list := config_list_header { config_list_entry };

config_list_header :=
    '[' 'RM-Cfg-List' ']';

config_list_entry :=
    config_list_name '=' { config_id ',' };

config_list_name := STRING(16);


config_active := config_active_header config_active_entry;

config_active_header :=
    '[' 'RM-Cfg-Active' ']';

config_active_entry :=
    'Active' '=' config_list_name;
```

3.2.3.3. Examples

The following configuration examples are from a working configuration file.

1. Service Configuration Items

These are the items that are available for activation and deactivation. In some cases, PIDs have been specified to deliver certain configuration to the item, e.g. to put it into 9600 bps mode, or specify timeslots to be used for the Tone Generator.

```
[RM-Cfg-Item-01]
Name=Utility - Monitor
Info=0020,FFFF
Pids=50,04,00,00,25,80   ; 9600

[RM-Cfg-Item-02]
Name=Fp Port 1 - Monitor
Info=0030,FFFF
Pids=50,04,00,00,25,80   ; 9600

[RM-Cfg-Item-03]
Name=FP Port 1 - Jump
Info=0031,FFFF
Pids=50,04,00,00,25,80   ; 9600

[RM-Cfg-Item-04]
Name=Ethernet - LAN
Info=0050,FFFF

[RM-Cfg-Item-05]
Name=Backplane - CCBUS
Info=0060,FFFF

[RM-Cfg-Item-06]
Name=Backplane - ToneGen
Info=0065,0010                     ; QMC0
Pids=A1,03,00,00,FF,A2,03,00,00,FF   ; Tx/Rx: 0 / Mask: FF

[RM-Cfg-Item-07]
Name=Fp Port 1 - Generator
Info=800C,FFFF
Pids=50,04,00,00,96,00   ; 38400
```

2. Service Configuration Lists

This contains three lists, *Multi*, *Release* and *Matthew*. If one of these lists are selected, then the specified Items will be activated. Note that any other additional Items are available to be activated and deactivated via. the Monitor.

```
[RM-Cfg-List]
Multi=04,05,06,02
Release=04,05,06,01,03
Matthew=04,05,06,01,03
```

3. Service Configuration Active

This specifies that the *Default* list is to be active. This is a special case, where if the software was compiled under MULTI then the *Multi* list is selected, otherwise if it is compiled without MULTI, the *Release* list is selected. Any other list name (e.g. *Matthew*) will refer directly to the named configuration list.

```
[RM-Cfg-Active]
Active=Default
```

To implement this configuration, the configuration file needs to be updated prior to execution of the operational image. As the image is starting, the Resource Management entities pass through an initiation phase in which they ask for, and receive, information from the configuration file.

3.2.4. Diagnostics

The Resource Manager, Service Manager and Connection Manager support a diagnostic level of reporting that can be specified in the configuration file. This can also be changed at run time through a message interface.

3.2.4.1. Description

The diagnostic level can be a value of 0 to 4 inclusive. 0 specifies None, 1 specifies Errors, 2 specifies Warnings and 3 specifies Informational, and 4 specifies Debugging.

3.2.4.2. Grammar

The following grammar is used to specify the level.

```
resource_general := resource_general_header { resource_item };
resource_item := reporting;

service_general := service_general_header { service_item };
service_item := reporting;

connection_general := connection_general_header { connection_item };
connection_item := reporting;

reporting :=
    'Reporting' '=' reporting_level;
reporting_level :=
    0 | 1 | 2 | 3 | 4;
```

3.2.4.3. Examples

The following configuration is used to set the RM, SM and CM reporting levels.

```
[RM-General]
Reporting=3      ; 4=debug

[SM-General]
Reporting=3      ; 4=debug

[CM-General]
Reporting=3      ; 4=debug
```

4. Diagnostics

Diagnostics are provided, or can be accessed, during the operation of the software. This provides the user with a means of observing and examining the state of the software. This can occur by setting reporting levels, or by interacting with the software via. the Monitor.

4.1. Reporting Levels

By enabling reporting levels, it is possible to have the Resource Management software provide a runtime display of its operational output.

Reporting Levels are available for the Resource Manager, Service Manager and Connection Manager. They can be enabled in any of three ways.

4.1.1. Configuration File Reporting Level Specification

The Reporting Level can be specified in the configuration file through a single statement. Refer to the previous section regarding Soft Configuration to examine the specification for this.

4.1.2. JEXEC_COMMAND Reporting Level Specification

The Reporting Level can be changed through the use of a JEXEC_COMMAND message. This conforms to the format as used by the Jexec Monitor.

The new Reporting Level is contained in parameter 0 of the message, and may have the value None (0x00), Error (0x01), Warn (0x02), Info (0x03) or Debug (0x04).

The message is sent to either the Resource Manager, Service Manager or Connection Manager.

4.1.3. JEXEC_STATUS_REPORT Reporting Level Specification

The Reporting Level can be changed through the use of a JEXEC_STATUS_REPORT message.

Parameter 0 of the message contains the value RM_DEBUG_REPORTING (0x01).

The new Reporting Level is contained in parameter 2 of the message, and may have the value None (0x00), Error (0x01), Warn (0x02), Info (0x03) or Debug (0x04).

The message is sent to either the Resource Manager, Service Manager or Connection Manager.

4.2. Monitor Access

By using commands available through the Monitor, it is possible to examine the internal state of, and to manipulate, the Resource Management software.

Monitor access is available for the Resource Manager, Service Manager, Connection Manager and Configuration Task. There are a number of commands that are available for each of these.

The Resource, Service and Connection Manager commands are accessed through the Monitor via. the "UP" prefix, whereas the Configuration commands are accessed through the Monitor via. the "US" prefix.

4.2.1. Resource Manager Commands

The commands provide access to Resources, Instances and Configuration. The purpose of this is to provide the user with information about which Resources have signed on, which Instances are in use, and what Configuration is available. There are additional commands that allow the user to Disconnect any Connected Instances.

The commands are accessed from monitor by using the 'R' prefix. Without any arguments, the following help information is shown.

```
Monitor>U P R
Command Reference:  r
  help -- Command Help Reference
  res [cmd] -- Resource SubCommands
  ins [cmd] -- Resource Instance SubCommands
  cfg [cmd] -- Resource Config SubCommands
  notify [cmd] -- Resource Notify SubCommands
  level 0|1|2|3|4 -- Specify Resource Manager Reporting Level
```

From this point, the subcommands can be used to access Resources, Instances, Configuration, Notifications and Reporting Levels. Use of these commands is self explanatory.

4.2.2. Service Manager Commands

The commands provide access to Services, Instances, Configuration and Users. The purpose of this is to provide the user with information about which Services have signed on, which Instances are in use, and what Configuration is available.

There are additional commands that allow the user to Disconnect any Connected Instances.

The commands are accessed from monitor by using the 'S' prefix. Without any arguments, help information is shown.

```
Monitor>U P S
Command Reference:  s
  help -- Command Help Reference
  ser [cmd] -- Service SubCommands
  ins [cmd] -- Service Instance SubCommands
  cfg [cmd] -- Service Config SubCommands
  user [cmd] -- Service User SubCommands
  level 0|1|2|3|4 -- Specify Service Manager Reporting Level
```

From this point, the subcommands can be used to access Services, Instances, Configuration, Users and Reporting Levels. Use of these commands is self explanatory.

4.2.3. Connection Manager Commands

These commands provide access to Connections, Nodes, Links and Users. The purpose of this is to provide the user with information about which Connections are in use.

The commands are accessed from monitor by using the 'C' prefix. Without any arguments, the following help information is shown.

```
Monitor>U P C
Command Reference:  c
  help -- Command Help Reference
  conn [cmd] -- Connection SubCommands
  node [cmd] -- Connection Node SubCommands
  user [cmd] -- Connection User SubCommands
  level 0|1|2|3|4 -- Specify Connection Manager Reporting Level
```

From this point, the subcommands can be used to access Connections, Nodes, Links and Reporting Levels. Use of these commands is self explanatory.

4.2.4. Configuration Manager Commands

These commands provide access to Service Items and Lists. These specify Services that are activated when the software starts, or via. the Monitor. The express purpose of this task is to provide a way of automatically activating Services that are not otherwise started by other tasks.

By using the '?' command, the following help is provided.

```
Monitor>U S ?
[00:00:00] rm_cfg: mon (): cmd( ? )
[00:00:00] rm_cfg: mon (): 'L (0..4)'          -- print set level
[00:00:00] rm_cfg: mon (): 'S'                 -- service show status
[00:00:00] rm_cfg: mon (): 'C'                 -- service config dump
[00:00:00] rm_cfg: mon (): 'A (id)'            -- service activate
[00:00:00] rm_cfg: mon (): 'D (id)'            -- service deactivate
[00:00:00] rm_cfg: mon (): 'I (id)'            -- service getinfo
Monitor>
```

From this point, the subcommands can be used to interrogate, and manipulate the state of the Services.

5. Applications

This section describes applications involving Resource Management.

5.1. Creating a Service

Services are created to fulfil an application requirement. A procedure to create one can take the following steps.

5.1.1. Construct the Service

Allocate a Service Identifier.

Any available Service Identifier can be chosen, they are listed in *rm_ser_n.h*. The Service Identifier should be put into *rm_ser_n.h* in the same manner as the existing Service Identifiers. This header file is stored in the common J5000 include directory.

Define & place the Service into the Configuration File.

The Service Definition is placed into one of the configuration files stored in the *config/cfg/jconfig* directory. This uses the formats as described above. The comments are stripped from the file before it is stored in the board, to conserve space. Even if the Service is hard coded, a minimal entry should be placed into the file.

Create & place the Service Class into the Software.

If a Service was defined through software modifications, then it needs to be placed into the build. This occurs by putting the software (body and header) into the *rm/sermgr/service* directory and ensuring that the *s_factry.h/.cpp* file is updated so that the Service is created when required.

5.1.2. Construct the Resources

Allocate a Resource Identifier.

The Resource Identifiers usually correspond to a JEXEC task id, but some Identifiers are pseudo Identifiers used to represent Hardware or other non-task entity (e.g. a virtual entity that acts as a token). The Identifier is placed into *rm_res_n.h* in the same manner that existing identifiers are stored. This header file is stored in the common J5000 include directory.

Define & place the Resource into the Configuration File.

The Resource Definition is placed into one of the configuration files stored in the *config/cfg/jconfig* directory. It uses the formats as described above. The comments are stripped from the file before it is stored in the board, to conserve space. An entry should be placed even if the Resource is hard coded.

Create & place the Resource Class into the Software.

If a Resource was defined through software modifications, then it needs to be placed into the build. This occurs by putting the software (body and header) into the *rm/resmgr/resource* directory and ensuring that the *r_factry.h/.cpp* file is updated so that the Resource is created when required.

5.1.3. Update the Configuration List

Create a Configuration Item.

If there is a need to activate the Service via. the Monitor, either at start up, or at the behest of a user, then a Configuration Item will need to be created. This takes the form as described above, and the item is placed into one of the configuration files stored in the *config/cfg/jconfig* directory.

Create or Update a Configuration List.

If there is a need to activate the Service when the system starts, then the Configuration Item should be placed into a Configuration List.

Modify the Active Entry.

The Configuration List, if not already defined as the Active List, should be made active it is required.

5.1.4. Update the System Configuration

The configuration changes will only take effect when the configuration is rebuilt and stored on the board.

5.1.5. Rebuild the Resource Management software

If modifications to software classes were made, then it is necessary to rebuild the software.

6. Example

As an example, the following illustrates the use of a simple configuration file, and interaction with the software.

The minimal configuration file contains a small number of Resources and Services.

```
[RM-General]
Reporting=3

[RM-Resource-00EE-00-RM]
Name=Sw Monitor
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-00A6-00-RM]
Name=Sw Rtrware
IIns=L,S,L,0,FFFFFFFF
IIns=U,S,U,0,FFFFFFFF
Mode=01,61,04,00,00,00,02
Mode=02,61,04,00,00,00,01
Mode=03,61,04,00,00,00,03

[RM-Resource-00F0-00-RM]
Name=Sw CCBus
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-00F2-00-RM]
Name=Sw JAsync
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-00ED-00-RM]
Name=Sw Terminal
IIns=L,S,L,0,FFFFFFFF
Mode=01,A0,01,03,A1,01,01,A2,01,05
Mode=01,A3,05,4C,6F,6F,70,00
Mode=02,A0,01,01,A1,01,01,A2,01,05
Mode=02,A3,05,47,65,6E,72,00,A4,04,54,65,73,74
Mode=02,A7,02,00,20,A8,02,02,00
Mode=03,A0,01,02,A1,01,01,A2,01,05
Mode=03,A3,05,54,65,72,6D,00
Mode=04,A0,01,01,A1,01,01,A2,01,05
Mode=04,A3,05,52,61,6E,64,00,A5,01,20
Mode=04,A7,02,00,20,A8,02,02,00
Mode=05,A0,01,03,A1,01,01,A2,01,05
Mode=05,A3,05,44,72,6F,70,00
Mode=05,A9,01,5F

[RM-Resource-00F4-00-RM]
Name=Sw Tone Generator
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-00F7-00-RM]
Name=DL Task

[RM-Resource-E002-00-RM]
Name=DL Sap Char
IIns=U,S,U,0,FFFFFFFF
```

```
[RM-Resource-E004-00-RM]
Name=DL Sap Packet
IIns=U,S,U,0,FFFFFFFF

[RM-Resource-E005-00-RM]
Name=DL Sap JMessage
IIns=U,S,U,0,FFFFFFFF

[RM-Resource-E008-00-RM]
Name=DL Stk Trans
Mode=00,40,04,00,00,00,0A,42,04,00,00,00,0A
Mode=01,40,04,00,00,00,F8,42,04,00,00,00,F8
Mode=02,40,04,00,00,05,FA,42,04,00,00,05,FA
Mode=03,40,04,00,00,10,00,42,04,00,00,10,00

[RM-Resource-E00A-00-RM]
Name=DL Mac Uart Scc
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-E010-00-RM]
Name=DL Mac Uart Smc
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-E00E-00-RM]
Name=DL Mac Ethernet
IIns=L,S,L,0,FFFFFFFF

[RM-Resource-E00F-00-RM]
Name=DL Mac QMC
IIns=L,S,L,0,FFFFFFFF
Mode=01,A0,04,00,00,00,01,A8,04,00,00,00,00
Mode=02,A0,04,00,00,00,00,A8,04,00,00,00,00
Mode=03,A0,04,00,00,00,81,A8,04,00,00,00,00
Mode=04,A0,04,00,00,00,80,A8,04,00,00,00,00
Mode=05,A0,04,00,00,00,00,A8,04,00,00,00,01

[RM-Resource-F001-00-RM]
Name=Hw Quicc Scc
IIns=U,S,U,0,FFFFFFFF

[RM-Resource-F002-00-RM]
Name=Hw Quicc Smc
IIns=U,S,U,0,FFFFFFFF

[RM-Resource-F004-00-RM]
Name=Hw Backplane
IIns=U,S,U,0,FFFFFFFF

[RM-Resource-F005-00-RM]
Name=Hw Quicc Qmc
IIns=U,S,U,0,FFFFFFFF


[SM-General]
Reporting=3

[SM-Service-0020]
Name=FP Port Utlty - Monitor
RIns=00,00EE,FF,00,00
RIns=01,E002,08,00,00
RIns=02,E008,08,00,00
RIns=03,E010,08,00,00
RIns=04,F002,00,00,00
```

```
RIns=05,00F7,08,00,00
CIns=00,00,L,01,U,0000
CIns=01,03,L,04,U,0000

[SM-Service-0031]
Name=FP Port 1 - Jump
RIns=00,00F2,FF,00,00
RIns=01,E002,02,00,00
RIns=02,E008,02,00,00
RIns=03,E00A,02,00,00
RIns=04,F001,02,00,00
RIns=05,00F7,02,00,00
CIns=00,00,L,01,U,0000
CIns=01,03,L,04,U,0000

[SM-Service-800C]
Name=FP Port 1 - Generator
RIns=00,00ED,FF,00,04
RIns=01,E005,02,00,00
RIns=02,E008,02,00,02
RIns=03,E00A,02,00,00
RIns=04,F001,02,00,00
RIns=05,00F7,02,00,00
CIns=00,00,L,01,U,0000
CIns=01,03,L,04,U,0000

[SM-Service-0060]
Name=Backplane - CCBus
RIns=00,00F0,FF,00,00
RIns=01,E005,01,00,00
RIns=02,E008,01,00,02
RIns=03,E00B,01,00,00
RIns=04,F001,01,00,00
RIns=05,00F7,01,00,00
CIns=00,00,L,01,U,0000
CIns=01,03,L,04,U,0000

[SM-Service-0065]
Name=Backplane - Tone Generator
RIns=00,00F4,FF,00,00
RIns=01,E005,10,00,00
RIns=02,E008,10,00,00
RIns=03,E00F,10,00,03
RIns=04,F005,10,00,00
RIns=05,00F7,10,00,00
CIns=00,00,L,01,U,0000
CIns=01,03,L,04,U,0000


[CM-General]
Reporting=3


[RM-Cfg-Item-01]
Name=Utility - Monitor
Info=0020,FFFF
Pids=50,04,00,00,25,80

[RM-Cfg-Item-02]
Name=Fp Port 1 - Monitor
Info=0030,FFFF
Pids=50,04,00,00,25,80
```

```
[RM-Cfg-Item-03]
Name=FP Port 1 - Jump
Info=0031,FFFF
Pids=50,04,00,00,25,80

[RM-Cfg-Item-04]
Name=Ethernet - LAN
Info=0050,FFFF

[RM-Cfg-Item-05]
Name=Backplane - CCBUS
Info=0060,FFFF

[RM-Cfg-Item-06]
Name=Backplane - ToneGen
Info=0065,0010
Pids=A1,03,00,00,FF,A2,03,00,00,FF

[RM-Cfg-Item-07]
Name=Fp Port 1 - Generator
Info=800C,FFFF
Pids=50,04,00,00,96,00


[RM-Cfg-List]
Multi=02
Release=01

[RM-Cfg-Active]
Active=Default
```

When started, the Monitor activates (it is defined as the first service).

```
Startup>run rm.gz
Loading File rm.gz
Startup>
Monitor Started (Stream 0).
Monitor>
```

The 'USS' command will show which Items are available, and their state.

```
Monitor>USS
[00:00:00] rm_cfg: mon (): cmd(s )
[00:00:00] rm_cfg: RM Config Status:
[00:00:00] rm_cfg:   Id  Name                   A ServId    State
[00:00:00] rm_cfg:   1   Utility - Monitor      Y 0020/0000 cncted
[00:00:00] rm_cfg:   2   Fp Port 1 - Monitor    N 0030/ffff idle
[00:00:00] rm_cfg:   3   FP Port 1 - Jump       N 0031/ffff idle
[00:00:00] rm_cfg:   4   Ethernet - LAN         N 0050/ffff idle
[00:00:00] rm_cfg:   5   Backplane - CCBUS      N 0060/ffff idle
[00:00:00] rm_cfg:   6   Backplane - ToneGen    N 0065/0010 idle
[00:00:00] rm_cfg:   7   Fp Port 1 - Generator  N 800c/ffff idle
Monitor>
```

It is possible to activate the 7th item, using the activate command.

```
Monitor>USA7
[00:00:00] rm_cfg: mon (): cmd(a7 )
[00:00:00] rm_cfg: mon (): activate: Fp Port 1 - Generator
[00:00:00] rm_cfg: Service Connecting: [7] Fp Port 1 - Generator
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - ResourceInstance (00ED/00/Sw Terminal)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - ResourceInstance (E005/02/DL Sap JMessage)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - ResourceInstance (E008/02/DL Stk Trans)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - ResourceInstance (E00A/02/DL Mac Uart Scc)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - ResourceInstance (F001/02/Hw Quicc Scc)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - ResourceInstance (00F7/02/DL Task)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - Connection:SWtoSW (0003)
[00:00:00] sm_sys: ServiceInstance (800C/00/FP Port 1 - Generator):
Created - Connection:SWtoSW (0004)
[00:00:00] rm_cfg: Service Connected: [7] Fp Port 1 - Generator
Data Terminal (): Resource Configure(0)
[00:00:00] da_trm:   Report Type = 1
[00:00:00] da_trm:   Report Time = 5 secs
[00:00:00] da_trm:   Generator   = Random
[00:00:00] da_trm:   Random      = (32 Length)
[00:00:00] da_trm:   Packet Size = 32 bytes
[00:00:00] da_trm:   Packet Time = 512 millisecs
Data Terminal (): Resource Connect(0)
Datalink (): Resource Configure(2)
Datalink (): Configure(Layer Sap, Object 5)
Datalink (): Resource Configure(2)
Datalink (): Configure(Layer Stack, Object 1)
St Tr(2): Configure
St Q(2): txBufSize=1530 txBufQua=0 rxBufSize=1530 rxBufQua=0
Datalink (): Resource Configure(2)
Datalink (): Configure(Layer Mac, Object 1, Device 2)
MacScc  Uart(2):  rBase=a4080500  tBase=a4080540  maxRxBD=8  maxTxBD=8
loopback=0
     DataRate=38400, DataBits=8, Parity=0, StopBits=1
     MaxRxBufferLen=1
Datalink (): Resource Connect(2)
Monitor>
```

By listing Resource Instances, it is possible to see that those are in the connected state -- including those that are part of the Service Instance just created.

```
Monitor>U P R INS SHOW
 Id    Copy  Name               State
 ----- ----- ------------------ ------------
 E00A  0002  DL Mac Uart Scc    Connected
 E010  0008  DL Mac Uart Smc    Connected
 E008  0002  DL Stk Trans       Connected
 E008  0008  DL Stk Trans       Connected
 E002  0008  DL Sap Char        Connected
 E005  0002  DL Sap JMessage   Connected
 00F7  0002  DL Task           Connected
 00F7  0008  DL Task           Connected
 00EE  0000  Sw Monitor        Connected
 F001  0002  Hw Quicc Scc      Connected
 F002  0000  Hw Quicc Smc      Connected
 00ED  0000  Sw Terminal       Connected
 ----- ----- ----------------- ------------

Monitor>
```

By listing Service Instances, it is possible to see those that are in the connected state, including those just created.

```
Monitor>U P S INS SHOW
 Id    Copy  Name                              State
 ----- ----- --------------------------------- ------------
 0020  0000  FP Port Utlty - Monitor           Connected
 800C  0000  FP Port 1 - Generator             Connected
 ----- ----- --------------------------------- ------------
Monitor>
```

The End.