

## Mercury Series Jukeboxes

### Navigation

[Table of Contents](#)

[Index](#)

[A - I](#)

[K - S](#)

[T - W](#)



**Author: Peter Gossler**  
**Department: JTST**  
**NSM Jukebox GmbH**

**Part. No. 141 208**

# Table of Contents

|  |            |
|--|------------|
| <b>Preface</b> .....                   | <b>I</b>   |
| Copyrights and Trademarks .....        | I          |
| Notice of Copyright .....              | I          |
| Disclaimer .....                       | I          |
| Changes .....                          | I          |
| Trademarks Used in this Manual .....   | II         |
| How to Use This Manual .....           | II         |
| Document Conventions .....             | II         |
| Points of Contact .....                | II         |
| Changer Model .....                    | III        |
| Robotics Interface Specification ..... | III        |
| Robotics Commands .....                | III        |
| Document Conventions .....             | IV         |
| Points of Contact .....                | V          |
| USA: .....                             | V          |
| World-Wide: .....                      | V          |
| WEB: .....                             | V          |
| <b>1 Changer Model</b> .....           | <b>1-1</b> |
| 1.1 General Description .....          | 1-1        |
| 1.2 Requirements for Operation .....   | 1-1        |
| 1.3 Security Aspects .....             | 1-1        |
| 1.3.1 Key-Lock .....                   | 1-2        |
| 1.3.2 Software Control .....           | 1-2        |
| 1.4 Characteristics .....              | 1-2        |
| 1.4.1 Command Processing .....         | 1-2        |
| 1.4.2 Command Examples .....           | 1-3        |
| 1.5 Unit Attention Conditions .....    | 1-6        |
| 1.6 Error Recovery .....               | 1-6        |
| 1.7 Logical Addressing Scheme .....    | 1-7        |
| 1.7.1 Drives .....                     | 1-7        |
| 1.7.2 Mail-slot .....                  | 1-7        |
| 1.7.3 Magazines & Trays .....          | 1-8        |

|          |  |            |
|----------|--|------------|
| 1.7.4    | Element Addressing .....   | 1-8        |
| <b>2</b> | <b>Robotics Interface Specification .....</b>  | <b>2-1</b> |
| 2.1      | Robotics Command and Data Format .....   | 2-1        |
| 2.2      | Robotics Command Descriptor Block (RCDB) .....   | 2-2        |
| 2.2.1    | ID Byte .....  | 2-2        |
| 2.2.2    | Operation Code .....   | 2-3        |
| 2.2.3    | Parameter/Data Bytes .....   | 2-3        |
| 2.2.4    | CRC16 .....  | 2-3        |
| 2.2.5    | RCDB Terminator .....  | 2-3        |
| 2.3      | Response Data Block (RDB) .....  | 2-4        |
| 2.3.1    | ID Byte .....  | 2-4        |
| 2.3.2    | Operation Code .....   | 2-4        |
| 2.3.3    | Command Execution Status (CES) .....   | 2-4        |
| 2.3.4    | Data Bytes .....   | 2-6        |
| 2.3.5    | CRC16 .....  | 2-6        |
| 2.3.6    | RDB Terminator .....   | 2-6        |
| <b>3</b> | <b>Operation Codes .....</b>   | <b>3-1</b> |
| 3.1      | Exchange Medium .....  | 3-2        |
| 3.2      | Initialize Element Status .....  | 3-3        |
| 3.3      | Inquiry .....  | 3-4        |
| 3.4      | Move Medium .....  | 3-8        |
| 3.5      | ReadSESW .....   | 3-9        |
| 3.6      | Request Device ID .....  | 3-15       |
| 3.7      | Request Exchange Status .....  | 3-17       |
| 3.7.1    | Request Exchange Status Command in Conjunction with the Exchange<br>Medium Command ..... | 3-20       |
| 3.8      | Request Status .....   | 3-23       |
| 3.9      | Rezero Unit .....  | 3-29       |
| 3.10     | UI Command .....   | 3-30       |
| 3.10.1   | UI Action Parameter Description. ....  | 3-31       |
| 3.11     | WriteSESW .....  | 3-36       |

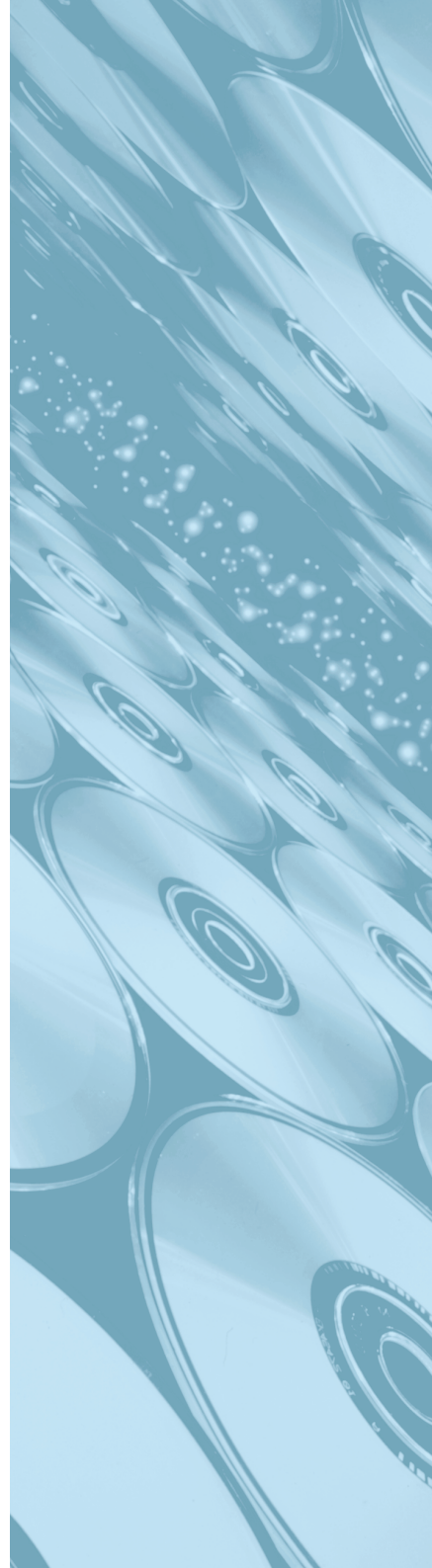
# Table of Contents

|             |     |
|-------------|-----|
| Index ..... | I-1 |
|-------------|-----|



# Preface

- **Copyrights and Trademarks**
- **Trademarks Used in this Manual**
- **How to Use This Manual**
- **Document Conventions**
- **Points of Contact**







## Preface

### Copyrights and Trademarks

#### Notice of Copyright

All rights reserved. Copyright © 1995-1997 by NSM Jukebox GmbH. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of **NSM Jukebox GmbH, 55411 Bingen, Im Tiergarten 20-30, Germany.**

#### Disclaimer

NSM JUKEBOX (NSM Jukebox GmbH and all of its subsidiaries) makes no representation or warranty with respect of the adequacy of this documentation or the procedures which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall NSM JUKEBOX, its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claim for lost profits, fees or expenses of any nature or kind.

#### Changes

The material in this specification is for information only and is subject to change without notice. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, NSM Jukebox GmbH assumes no liability resulting from any errors or omissions in this manual, or from the use of the information contained herein.

NSM Jukebox GmbH reserves the right to make changes in the product design without reservation and without notification to its users.

# Preface

## How to Use This Manual

Additional Information may be obtained from:

**NSM Jukebox GmbH**  
**Department JTST**  
**55411 Bingen am Rhein**  
**Im Tiergarten 20-30**  
**Germany**

## Trademarks Used in this Manual

Adaptec is a registered trademark of Adaptec, Inc.

Intel is a registered trademark of Intel Corp.

Microsoft is registered trademark of Microsoft Corp.

MS-DOS is a registered trademark of Microsoft Corp.

Novell is a registered trademark of Novell, Inc.

NetWare is a registered trademark of Novell, Inc.

NSM Jukebox is a registered trademark of NSM Jukebox GmbH

Windows, Windows 95 and Windows NT are registered trademarks of Microsoft Corp.

## How to Use This Manual

Turn to the chapter in this manual that contains the information you need.

### Document Conventions

This chapter describes the conventions used in this document. Read this chapter if you are not familiar with the conventions used in this documentation.

### Points of Contact

This chapter describes contact addresses of NSM Jukebox. Read this chapter if you are searching for a contact address of NSM Jukebox next to you.

## Changer Model

This chapter describes the model of the medium changer. Turn to this chapter if you are not familiar with the general organization, function and features of the **Mercury** product line.

## Robotics Interface Specification

This chapter describes the interface capabilities and features of the **Mercury** product line. Turn to this chapter if you are not familiar with the **Mercury** command format specifics.

## Robotics Commands

This chapter describes the supported commands and the appropriate responses of the **Mercury**. Turn to this chapter if you are looking for command details and how to implement the commands into your code.

## Document Conventions

The following conventions are used throughout this manual to define syntax:

Table 1: Document Conventions

| Convention                     | Meaning  |
|--------------------------------|--|
| <b>Bold text</b>               | Denotes keywords, proper names or words that will be explained further on in the text or <b>Glossary</b> .   |
| <i>Italic text</i>             | Denotes a place holder or variable you have to provide. For example: the statement <i>Move Medium (X, Y)</i> requires you to substitute values for the <i>X</i> and <i>Y</i> parameters. The statement <i>Move Medium (0101, 0001)</i> requires you to substitute the values exactly as specified. |
| <b><i>Bold Italic text</i></b> | Denotes cross-references to Chapters, Tables, Figures, etc.  |
| <b>FULL CAPITALS</b>           | Denotes file names.  |
| Display Text                   | Denotes messages that will be shown on the LC-display of the jukebox.  |
| COMMAND                        | Denotes robotics command names.  |
| 'Number'                       | Numbers in <b>quotation marks</b> identify <b>ASCII</b> values. For example: '4' represents the hex value 34h. '345' represents a string holding the characters 3, 4 and 5 (0x33, 0x34 and 0x35).  |
| 0xNumber                       | Numbers lead by <b>0x</b> are <b>hexa-decimal</b> values. For example: the string <b>0x10</b> represents the decimal value 16.   |
| Number                         | Numbers without preceding 0x or in quotation marks represent <b>decimal</b> values.  |

## Points of Contact

### USA:

NSM Jukebox of America  
1158 Tower Lane  
Bensenville, Illinois 60106  
Phone: +1 (630) 860-5100  
Fax: +1 (630) 860-5144  
Email: ddanoski@nsmjukebox.com

### World-Wide:

NSM Jukebox GmbH  
Im Tiergarten 20-30  
D-55411 Bingen am Rhein  
Phone: +49 (0)6721 964-430  
Fax: +49 (0)6721 964-414  
Email: pgossler@nsmjukebox.com

### WEB:

<http://www.nsmjukebox.com/p3/techsupp.html>

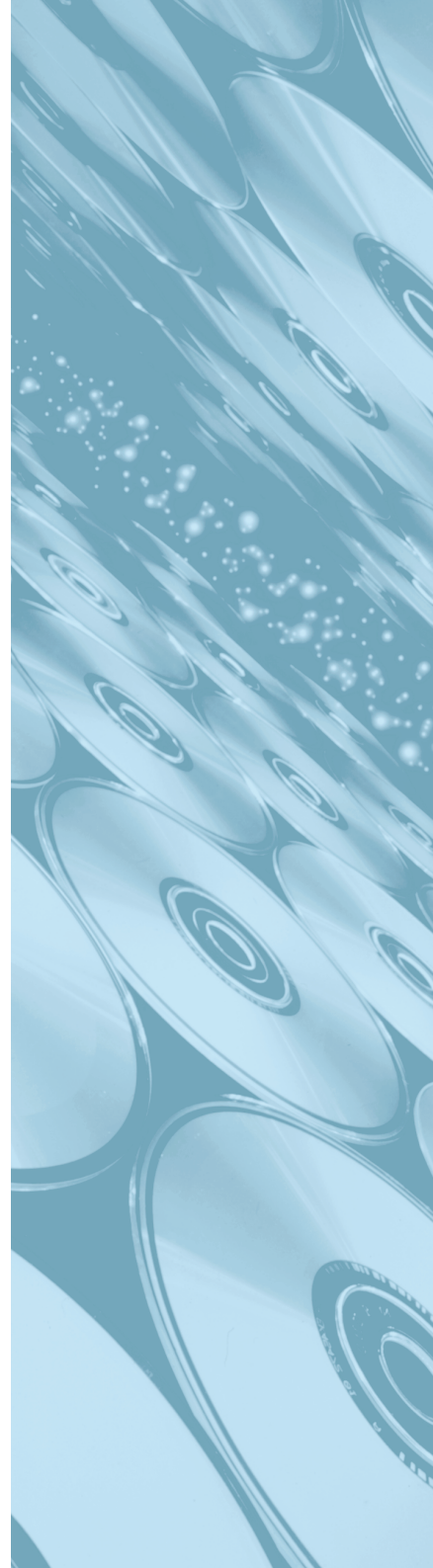


# Preface

## Points of Contact

# Changer Model

- General Description
- Requirements for Operation
- Security Aspects
- Characteristics
- Unit Attention Conditions
- Error Recovery
- Logical Addressing Scheme







# 1 Changer Model

## 1.1 General Description

The **Mercury** is a very fast CD-ROM changer that can hold up to 150 CDs and up to four CD-ROM drives. The 150 CDs are stored on trays residing in three magazines with a total capacity of 50 CDs each.

The **Mercury** provides an Import/Export element (Mail-slot) which allows to insert or withdraw a CD-ROM into/from the **Mercury**. This process can either be initiated via key-pad or host software.

Two hinged doors at the side of the **Mercury** provide access to the three magazines. It is very easy to exchange single magazines or even a set of magazines against another set of magazines through these doors.

## 1.2 Requirements for Operation

The **Mercury** always has to be operated fully populated with three magazines each holding 50 trays. The trays do not necessarily have to hold media. There is a possibility of mail-functioning if the **Mercury** is operated when trays are manually (not using the Mail-slot) removed from the magazines.

To ensure that an operator always has access to all 50 CDs in a magazine when swapping magazines, the **Mercury**, before opening the doors, unloads all drives and the Mail-slot and returns the mounted CDs to their home positions.

## 1.3 Security Aspects

As already mentioned above the media stored in the **Mercury** might be removed or exchanged using the Mail-slot or exchanging magazines. This does involve some security aspects relating to a host application operating the **Mercury**.

To ensure that no unwanted access to a medium that currently is mounted to a drive can occur the **Mercury** has two levels of security.

### 1.3.1 Key-Lock

The key-lock enables or refuses access to the **Open-Door** button and the **Key-Pad** of the **Mercury**. Once the operator has inserted the key into the key-lock and unlocked it, he has access to the menu and the door button of the **Mercury**. He then is allowed to open the doors and insert or withdraw CDs through the Mail-slot. The action he takes in this particular case cannot be monitored nor controlled by the host application software. Using the **Open-Door** button is defined as device maintenance.

### 1.3.2 Software Control

In this particular case the operator who wants to access the jukebox is not in possession of the **Mercury** key. Once the operator presses the Open-Door key, the host application will be notified by an UNIT ATTENTION condition (xxx) with the next command it sends to the **Mercury**. After notification the host application is responsible to clear all user links (user connections). After clearing all user links the application should issue a REZERO UNIT command (xxx) to instruct the **Mercury** to unload all drives, the Mail-slot and to open the doors.

## 1.4 Characteristics

The **Mercury** firmware is designed to minimize robotics movement and therefore to increase reliability. For you, as an implementor, it is necessary to understand the command processing of the **Mercury** more detailed.

### 1.4.1 Command Processing

The **Mercury** has a real-time, multi-threading operating system and is capable of asynchronous command processing. The **Mercury** does not have a command queue thus it will accept any command at any time unless there is a failure or the **Mercury** is busy. In the case of failure or a busy status the **Mercury** will only accept a sub-set of all supported commands (xxx for more details).

All actions taken by the **Mercury** are executed through threads. Some of the threads are active even in idle state (no command has to be proceeded) of the **Mercury**. For a better understanding of the **Mercury** some of these threads are listed below.

- **Command processor thread:**  
This thread waits for a command to be proceeded. Depending on the command, the command processor thread creates or starts sub-threads to fulfill the desired action or sets event variables to signal other threads to start.
- **Status update thread:**  
These threads constantly update the current status values of each element of the Mercury (e.g. *drive 0001-0004 status*, *Mail-slot status* etc.). The status values may be read by using the REQUEST STATUS command (xxx).
- **Move thread:**  
This thread starts several sub-threads that take care of all necessary actions to perform the movement of a disc from one location to another location. The move thread is started as soon as the command processor thread writes a new value into an event variable. This process is defined as **assigning a status** to a given target element (drive, Mail-slot or storage location). As long as the assigned state does not equal the current state, the process has not ended.
- **Termination thread:**  
This thread is used to terminate another active thread.

To explain the cooperation of some of the threads below is an example of a MOVE MEDIUM command.

### 1.4.2 Command Examples

#### A Single MOVE MEDIUM Command

Issued by an host application the **Mercury** retrieves a MOVE MEDIUM command. The MOVE MEDIUM command first triggers the **command processor thread** which validates the command and its parameters. If the command is valid the command processor thread writes the assigned state, given by the command parameters, to the proper event variable of the addressed target element.

The **command processor thread** triggers a thread that is responsible to return a command acknowledgment (**Response Data Block (RDB)**) to the host application.

The change in the contents of the event variable signals the **move thread** to start.

While all threads are working, **status update threads** constantly update the status variables of all elements so, that an application can retrieve the current machine status of the **Mercury** at any time.

Once the current element status equals the assigned element status for the given target element the MOVE MEDIUM command has successfully ended and the machine returns back to idle state if no further command processes are active.

### **MOVE MEDIUM Command Overlapping a Previous MOVE MEDIUM Command**

Let's assume the same scenario as described above thus, during command execution the host application sends a second MOVE MEDIUM command addressing the same target element with a different source element address.

The **command processor thread** triggers a thread that is responsible to return a command acknowledgment to the host application.

At the same time the change in the contents of the event variable signals a **move thread** with the newly assigned destination address to start.

The **command processor thread** triggers the **termination thread** to terminate all threads still active to achieve the status of the addressed element assigned by the previous MOVE MEDIUM command. The termination of the threads results in the start of a **move thread** that is responsible to return the medium addressed by the previous command to its home storage location.

While all threads are working, status update threads constantly update the status variables of all elements so, that an application can retrieve the current machine status of the **Mercury** at any time.

Once the current element status equals the assigned element status for the given target element the second MOVE MEDIUM command

has successfully ended and the machine returns back to idle state if no further command processes are active.

### **A MOVE MEDIUM Command Addressing a Target Element Holding a Medium**

Let's assume the same scenario as described under "*A Single Move Medium Command*" on page -3, except the target element already holds a medium.

The **command processor thread** triggers a thread that is responsible to return a command acknowledgment to the host application.

The change in the contents of the event variable creates a **move thread** that will automatically return the medium residing in the addressed destination element to its home storage location.

At the same time the change in the contents of the event variable signals the **move thread** to start.

While all threads are working, **status update threads** constantly update the status variables of all elements so, that an application can retrieve the current machine status of the **Mercury** at any time.

Once the current element status equals the assigned element status for the given target element the second MOVE MEDIUM command has successfully ended and the machine returns back to idle state if no further command processes are active.

### 1.5 Unit Attention Conditions

Whenever an unexpected condition exists the **Mercury** will notify an application by setting an **UNIT ATTENTION** condition (**UAC**) in the **Command Execution Status (CES)** of the **RDB** of the next command that it receives. In this case the command will not be processed by the **Mercury** and the host application will have to re-issue the same command again. It is important to understand, that several UACs may exist at the same time in the **Mercury** (e.g. **POWER ON RESET OCCURRED** and **DOOR CLOSED**). For every existing UAC the **Mercury** will notify the host application. This means that a command can be rejected several times via UAC. Only if all the UACs have been cleared by transmitting them to the host application, the **Mercury** will proceed the next command.

### 1.6 Error Recovery

The **Mercury** is designed to be very reliable and therefore has an intelligent error recovery mechanism build into its firmware. Whenever the **Mercury** returns a **FATAL FAILURE** (non-recoverable failure) it has already tried to recover the particular failure with its built in error recovery routines. In this case the host application should follow the procedure described below.

- Clear the failure condition by issuing a **REZERO UNIT** command with the Action Parameter set to **0x00** (see "Rezero Unit" on page 3.9-29).
- Re-issue the command reporting the failure again.
- If the command still reports a failure condition send a **REZERO UNIT** command with the Action Parameter set to **0x01** (see "Rezero Unit" on page 3.9-29), this will instruct the **Mercury** to clear the failure condition, unload all drives and the Mail-slot.
- Re-issue the command reporting the failure again.

- If the command still reports a failure condition send a REZERO UNIT command with the Action Parameter set to 0x02 (see "Rezero Unit" on page 3.9-29) this will instruct the **Mercury** to clear the failure condition, unload all drives and the Mail-slot and to open the doors of the **Mercury** so, that an operator has access to the **Mercury**.

## 1.7 Logical Addressing Scheme

In this chapter the physical to logical mapping of the **Mercury** Jukebox is shown.

The **Mercury** is divided into four logical sections; **section 00 to section 03**. To each of the sections elements (drives, Mail-slot, magazines) are assigned.

Each section has a 2 character base address and each element within the section has a 2 character element address. The building block of an address therefore always looks like the following:

**BBEE**

where **BB** = base address and **BB Element of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

and **EE** = element address and **EE Element of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

### 1.7.1 Drives

The drives of the **Mercury** are mounted in a staple originated in section 00. They are sub-sequentially numbered from **1 to 4**. The bottom most drive within the drive staple has logical address '**01**' and the top most drive of the drive stack has the logical address '**04**'. When addressing drive number **2** the address has to read '**0002**'.

### 1.7.2 Mail-slot

The Mail-slot has been assigned **address '0000'**.

### 1.7.3 Magazines & Trays

Each magazine has a staple of 50 CDs. A CD is addressed through its logical magazine number **BB** and its physical position within the magazine (tray) **EE**. The CDs are sub-sequentially numbered from **1 to 50** from **bottom to top** of a magazine. When addressing **CD 43** in magazine **2** the address has to be **'0243'** (**BBEE**).

### 1.7.4 Element Addressing

An element is always referenced by its current location. To move a CD to a drive and then back to its home location the following command sequence should be issued:

```
Move Medium ('0101' '0001') move disc '01', magazine  
'01' to drive '0001'
```

```
Move Medium ('0001' '0101') return the disc in drive  
'0001' to its home location.
```

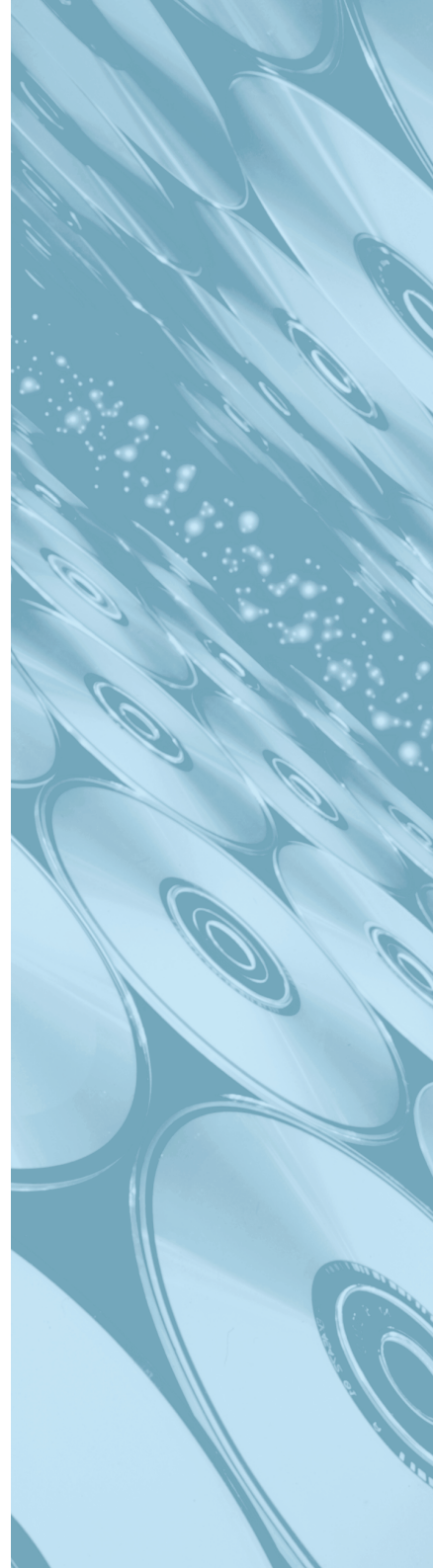
#### NOTE:

It is not possible to return a disc to a location other than its home location. Therefore, when returning a disc to its home location, the second parameter in the `MOVE MEDIUM` command is ignored.

If a disc is loaded into a drive or the Mail-slot, the application may move the disc directly from one element to another element without returning it to its home location first. The command for this procedure would use the home location address of the storage element to move as the source address parameter in the `MOVE MEDIUM` command and the address of the element where it should be moved to as the destination address parameter in the `MOVE MEDIUM` command. The **Mercury** then will unload the source element (drive) and move the storage element directly to the next destination element. This procedure minimizes lift movements.

# Robotics Interface Specifications

- **Robotics Command and Data Format**
- **Robotics Command Descriptor Block (RCDB)**
- **Response Data Block (RDB)**





## 2 Robotics Interface Specification

The communication to the robotics part of the **Mercury** has to be performed using a standard RS-232 interface.

The **Mercury** supports a **V24 NULL modem connection** through a 9 pin male D-Sub connector using the lines **RxD**, **TxD** and **SGND**. There is no support of any Hand-Shake signal. The robotics interface supports 9600 Baud, 8 Data bits, 1 Stop bit, No parity. The Jukebox does not support any other Baud-rate than 9600 Baud.

The serial interface of the **Mercury** has an input impedence that allows to daisy chain up to **3 Mercury**'s at a single serial host interface. The load on the serial interface with **3 Mercury** Jukeboxes is the same load as the load of one standard serial device (e.g. a modem).

A command is transmitted to the **Mercury** by sending a command descriptor block to the robotics controller (see "*Robotics Command Descriptor Block (RCDB)*" on page 2).

The command execution status (see "Response Data Block (RDB)" on page 4) returned in the response data block holds information if the command has been accepted if the command has been rejected.

### 2.1 Robotics Command and Data Format

The following section describes the format of commands sent to the **Mercury** as well as the responses returned by the **Mercury** in more detail.

A command assigns a status to the **Mercury**. The change of the assigned status value starts a process within the **Mercury** that is responsible to reach the assigned state e.g., moving a disc from A to B.

If, during command execution, the power fails and the assigned state could not be reached, after power is back up again, the **Mercury** robotics controller is in an undefined state. It will try to reach a valid state, e.g. returning a disc, still residing the lift, to its home position.

# Robotics Interface Specification

## Robotics Command Descriptor Block (RCDB)

For all commands, if there is an invalid parameter in the command descriptor block, then the robotics controller terminates the command without altering media.

### 2.2 Robotics Command Descriptor Block (RCDB)

Each robotics command sent to the controller will be acknowledged by the controller within  $t_{RSPMAX} = 5s$  with a **Response Data Block (RDB)**. If the RDB is **not** sent within  $t_{RSPMAX}$  there may be a connection or hardware problem. To see if there really is a problem, the command has to be re-transmitted until the robotics controller replies or a definite hardware problem can be achieved. The gap between two characters send over the RS-232 port is defined with  $t_{GAPMAX} = 1s$ . **If the gap between two subsequent characters is bigger then  $t_{GAPMAX}$**  there may be a connection or hardware problem.

Each robotics command shall have the following format:

**Table 2: Typical Robotics Command Descriptor Block**

| Bit Byte | 7                      | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------------|---|---|---|---|---|---|---|
| 0        | ID (hex)               |   |   |   |   |   |   |   |
| 1        | Operation Code         |   |   |   |   |   |   |   |
| 2        | n Parameter/Data Bytes |   |   |   |   |   |   |   |
| 2+n      | CRC16 (MSB)            |   |   |   |   |   |   |   |
| 3+n      | CRC16                  |   |   |   |   |   |   |   |
| 4+n      | CRC16                  |   |   |   |   |   |   |   |
| 5+n      | CRC16 (LSB)            |   |   |   |   |   |   |   |
| 6+n      | RCDB Terminator (0x00) |   |   |   |   |   |   |   |

#### 2.2.1 ID Byte

The **ID Byte** specifies the ID of the addressed **Mercury**, that is to be addressed. Valid ID values are 0x80 to 0x8F. The address

range of 0 to 15 allows up to 16 **Mercury** in a Daisy Chain..

### 2.2.2 Operation Code

The **Operation Code** is a code for the desired action. The operation code is defined by one ASCII character. Valid values are 20 to 127.

### 2.2.3 Parameter/Data Bytes

A number of fields holding the parameters or data needed by the robotics to perform the desired operation. Valid values for these parameters are 30 to 127.

### 2.2.4 CRC16

Four Bytes of cyclic redundancy checksum, valid values are '0'..'9' and 'A'..'F'. The 16 bit CRC-CCITT algorithm<sup>1)</sup>, shown in Equation 1, is used to calculate the checksum.

#### Equation 1: CRC-CCITT Checksum

$$\text{CRC} = x^{16} + x^{12} + x^5 + 1$$

Where:

- **CRC = Cyclic Redundancy Check**
- **x = Byte to be calculated**

### 2.2.5 RCDB Terminator

Byte that terminates the Robotics Command Descriptor Block, always **0x00**.

---

1. )Further information as well as programming samples and a description of the CRC-CCITT algorithm can be obtained from the NSM Jukebox JSDK (Jukebox Software Developers Kit).

### 2.3 Response Data Block (RDB)

Each **Response Data Block** returned from the robotics will have the following format:

Table 3: Typical Response Data Block (RDB)

| Bit Byte | 7  | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|--|---|---|---|---|---|---|---|
| 0        | ID (hex)                                   |   |   |   |   |   |   |   |
| 1        | Operation Code causing RDB                 |   |   |   |   |   |   |   |
| 2        | Command Execution Status Information (MSB) |   |   |   |   |   |   |   |
| 3        | Command Execution Status Information (LSB) |   |   |   |   |   |   |   |
| 4        | n Data Bytes                               |   |   |   |   |   |   |   |
| 4+n      | CRC16 (MSB)                                |   |   |   |   |   |   |   |
| 5+n      | CRC16                                      |   |   |   |   |   |   |   |
| 6+n      | CRC16                                      |   |   |   |   |   |   |   |
| 7+n      | CRC16 (LSB)                                |   |   |   |   |   |   |   |
| 8+n      | RDB Terminator (0x00)                      |   |   |   |   |   |   |   |

#### 2.3.1 ID Byte

The **ID Byte** identifies the ID of the **Mercury**, that sent the RDB. Possible ID values are 0xC0 through 0xCF.

#### 2.3.2 Operation Code

The **Operation Code** of the command that caused the robotics controller to return this RDB. The operation code consists of an one character ASCII field.

#### 2.3.3 Command Execution Status (CES)

The **Command Execution Status** (CES) field holds 2 characters of status information on the command received. The command execution status format (see **Table 4**) is ASCII.

**NOTE:**

The **Mercury** will immediately after checking the command syntax and parameters return the RDB. a Command Execution Status of '00' in the ces rather indicates that the command has been accepted and now is going to be executed than command completion

**Table 4: Command Execution Status Codes**

|      |                                 |
|------|---------------------------------|
| '0x' | Command Group                   |
| '00' | Command accepted                |
| '01' | Command unknown                 |
| '1x' | Invalid Command Parameter Group |
| '11' | Invalid command parameter count |
| '12' | Invalid command parameter value |
| '2x' | Busy Condition                  |
| '3x' | Failure Group                   |
| '4x' | Unit Attention Group            |
| '40' | Power On Reset Occurred.        |
| '42' | Media Changed                   |
| '43' | Operator Door Open Request      |
| '44' | Reserved                        |
| '45' | Door Closed                     |

**NOTE:**

Returning an UNIT ATTENTION status the machine tries to get the attention of a host application to communicate a specific condition which might not be expected by the host.

When a UNIT ATTENTION condition is reported, the command will not be proceeded and no media will be altered. The host has to re-issue the command.

A **Command Execution Status** not equal to '00' indicates that this command has been rejected by the **Mercury** .

A BUSY ('20') or FAILURE CONDITION ('30') indicates this condition has already been existing and has not been caused by the current command.

## Robotics Interface Specification

### Response Data Block (RDB)

POWER ON RESET OCCURRED ('40') indicates that the machine has been powered off and on again.

Exporting a reserved medium through the Mail-slot will result in a RESERVATION CONFLICT ('41') status.

MEDIA CHANGED ('42') status indicates that the addressed medium has been exchanged either through the Mail-slot or by opening the doors of the **Mercury**.

OPERATOR DOOR OPEN REQUEST ('43') indicates that the operator pressed the door open button at the front panel of the **Mercury** and wants to access the machine.

DOOR CLOSED ('45') indicates that the operator has closed the door.

#### 2.3.4 Data Bytes

A number (**n**) of Bytes holding the desired information. Usually this information is in ASCII format. The length of the data bytes returned may vary depending on the machine type as well as equipment of the machine. The actual RDB datar length must be determined as described in *Equation 2*.

#### Equation 2: RDB Data Length

$$\text{DataLen} = \text{Ptr2RDBTrm} - 4 - (\text{Ptr2CES} + 2)$$

Where:

- **DataLen** = Data Length
- **Ptr2RDBTrm** = Pointer to RDB Terminator Byte
- **Ptr2CES** = Pointer to first CES Byte

#### 2.3.5 CRC16

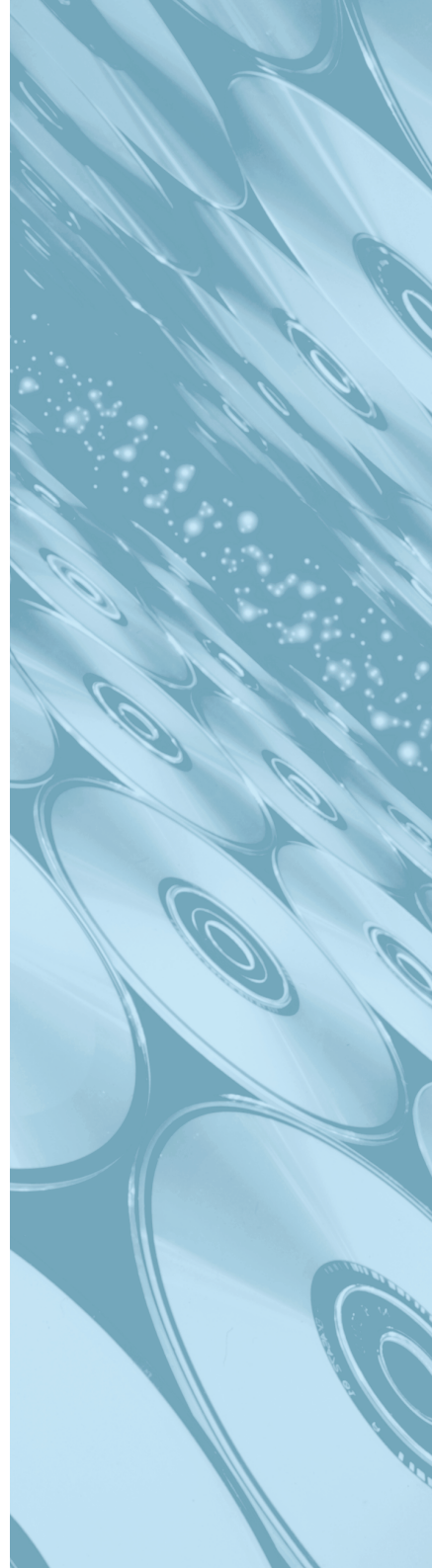
Four Bytes of **Cyclic Redundancy Checksum**, valid values are '0'..'9' and 'A'..'F'.

#### 2.3.6 RDB Terminator

Byte that terminates the Robotics Data Return Block. The RDB Terminator is always **0x00**.

# Operation Codes

- Exchange Medium
- Initialize Element Status
- Inquiry
- Move Medium
- ReadSESW
- Request Device ID
- Request Exchange Status
- Request Status
- Rezero Unit
- UI Command
- WriteSESW







### 3 Operation Codes

*Table 5* defines the robotics operation codes in ascending, alphabetical order.

**Table 5: List of supported commands**

| Command Name              | Op-Code |
|---------------------------|---------|
| EXCHANGE MEDIUM           | 0x25    |
| INITIALIZE ELEMENT STATUS | 0x26    |
| INQUIRY                   | 0x21    |
| MOVE MEDIUM               | 0x24    |
| READ SESW                 | 0x27    |
| REQUEST EXCHANGE STATUS   | 0x2B    |
| REQUEST DEVICE ID         | 0x2C    |
| REQUEST STATUS            | 0x22    |
| REZERO UNIT               | 0x20    |
| UI                        | 0x30    |
| WRITE SESW                | 0x2A    |

M40IFSpec3.fm / 09.02.1997/20.06.1997  
Document No. 141 208

### 3.1 Exchange Medium

The EXCHANGE MEDIUM command provides a means to exchange the medium in the first element address with the medium located at a second element address.

Table 6: EXCHANGE MEDIUM Command

| Bit Byte | 7                     | 6                      | 5 | 4 | 3 | 2 | 1 | 0     |
|----------|-----------------------|------------------------|---|---|---|---|---|-------|
| 0        | ID (hex)              |                        |   |   |   |   |   |       |
| 1        | Operation Code (0x25) |                        |   |   |   |   |   |       |
| 2        | (MSB)                 | First Element Address  |   |   |   |   |   |       |
| 5        |                       |                        |   |   |   |   |   | (LSB) |
| 6        | (MSB)                 | Second Element Address |   |   |   |   |   |       |
| 9        |                       |                        |   |   |   |   |   | (LSB) |
| 10       | (MSB)                 | CRC16                  |   |   |   |   |   |       |
| 13       |                       |                        |   |   |   |   |   | (LSB) |
| 14       | RDB Terminator (0x00) |                        |   |   |   |   |   |       |

The medium in the second element address is moved to the first element address and the medium which previously occupied the first element address is moved to the second element address.

The **First Element Address** field holds a 4 character string identifying the address of the first storage element that is to be exchanged. The format of the string is **mmtt** whereby **mm** = **magazine address** and **tt** = **slot address**, e.g. '0108' means magazine address 1 and slot address 8.

The **Second Element Address** field holds a 4 character string identifying the address of the second storage element which has to be exchanged with the medium specified by the first element address field. The format of the string is **mmtt** whereby **mm** = **magazine address** and **tt** = **slot address**, e.g. '0350' means magazine address 3 and slot address 50.

Only exchanges between storage elements are supported by the **Mercury**. If an application sends a command specifying a data transfer element in the first or second element address field, the command will be terminated with the command execution status field set to INVALID COMMAND PARAMETER VALUE.

### 3.2 Initialize Element Status

The INITIALIZE ELEMENT STATUS command will cause the **Mercury** to scan all storage elements for media. The intend of this command is to get a quick response from a following READSESW command. It may be useful to issue this command after a power failure, or if a medium has been exchanged by an operator.

Table 7: INITIALIZE ELEMENT STATUS Command

| Bit<br>Byte | 7                     | 6     | 5 | 4 | 3 | 2 | 1 | 0     |
|-------------|-----------------------|-------|---|---|---|---|---|-------|
| 0           | ID (hex)              |       |   |   |   |   |   |       |
| 1           | Operation Code (0x26) |       |   |   |   |   |   |       |
| 2           | (MSB)                 | CRC16 |   |   |   |   |   |       |
| 5           |                       |       |   |   |   |   |   | (LSB) |
| 6           | RDB Terminator (0x00) |       |   |   |   |   |   |       |

### 3.3 Inquiry

The INQUIRY command supports information describing the type of the **Mercury** and vital product data about the equipment of the **Mercury**.

**Table 8: INQUIRY Command**

| Bit Byte | 7                     | 6     | 5 | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|-------|---|---|---|---|---|-------|--|
| 0        | ID (hex)              |       |   |   |   |   |   |       |  |
| 1        | Operation Code (0x21) |       |   |   |   |   |   |       |  |
| 2        | (MSB)                 | CRC16 |   |   |   |   |   |       |  |
| 5        |                       |       |   |   |   |   |   | (LSB) |  |
| 6        | RDB Terminator (0x00) |       |   |   |   |   |   |       |  |

As a result to the INQUIRY command the jukebox will return the data described in *Table 9*.

**Table 9: INQUIRY RDB**

| Bit Byte | 7                     | 6                            | 5 | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|------------------------------|---|---|---|---|---|-------|--|
| 0        | ID (hex)ID            |                              |   |   |   |   |   |       |  |
| 1        | Operation Code (0x21) |                              |   |   |   |   |   |       |  |
| 2        | (MSB)                 | CommanID ID Execution Status |   |   |   |   |   |       |  |
| 3        |                       |                              |   |   |   |   |   | (LSB) |  |
| 4        | (MSB)                 | Vendor Identification        |   |   |   |   |   |       |  |
| 11       |                       |                              |   |   |   |   |   | (LSB) |  |
| 12       | (MSB)                 | Product Identification       |   |   |   |   |   |       |  |
| 27       |                       |                              |   |   |   |   |   | (LSB) |  |
| 28       | (MSB)                 | Product Revision Level       |   |   |   |   |   |       |  |
| 31       |                       |                              |   |   |   |   |   | (LSB) |  |
| 32       | (MSB)                 | Production Date              |   |   |   |   |   |       |  |

**Table 9: INQUIRY RDB**

|            |       |   |
|------------|-------|---|
| 39         |       | (LSB)                                       |
| 40         | (MSB) | Part Number                                 |
| 47         |       | (LSB)                                       |
| 48         | (MSB) | Serial Number                               |
| 59         |       | (LSB)                                       |
| 60         | (MSB) | Robotics Controller Firmware Release        |
| 63         |       | (LSB)                                       |
| 64         | (MSB) | Robotics Controller Firmware Date           |
| 71         |       | (LSB)                                       |
| 72         | (MSB) | Logical Position of Import Export Element   |
| 75         |       | (LSB)                                       |
| 76         | (MSB) | Maximum Magazine Count                      |
| 77         |       | (LSB)                                       |
| 78         | (MSB) | Maximum Media Count                         |
| 79         |       | (LSB)                                       |
| 80         | (MSB) | Data Transfer Element Count (N)             |
| 81         |       | (LSB)                                       |
| 82         | (MSB) | Logical Position of Data Transfer Element 1 |
| 85         |       | (LSB)                                       |
| 86+(N-1)*4 | (MSB) | Logical Position of Data Transfer Element N |
| 89+(N-1)*4 |       | (LSB)                                       |
| 90+(N-1)*4 | (MSB) | CRC16                                       |
| 93+(N-1)*4 |       | (LSB)                                       |
| 94+(N-1)*4 |       | RDB Terminator (0x00)                       |

M40IFSpec3.fm / 09.02.1997/20.06.1997  
Document No. 141\_208

The **Vendor Identification** field holds an 8 character string identifying the vendor of the Media Changer. OEM customers may implement their vendor identification herein. Per default this field contains the string ‘NSM’ and is filled with blanks (**0x20**).

The **Product Identification** field holds a 16 character string identifying the product itself. OEM customers may implement their product identification herein. Per default this field contains the string ‘Mercury-40’ and is filled with blanks (**0x20**).

The **Product Revision Level** field holds a 4 character string specifying the product revision level with four digits e.g. ‘1.51’.

The **Production Date** is an 8 character field specifying the date of production. The date format is **DD/MM/YY**.

The **Part Number** field holds an 8 character string specifying the part number of the product. OEM customers may implement their part number herein.

The **Serial Number** field holds a 12 character string specifying the serial number of the unit. The serial number is stored herein during production of the unit.

The **Robotics Controller Firmware Release** holds a 4 character string specifying the firmware revision of the robotics controller board, e.g. ‘0102’.

The **Robotics Controller Firmware Date** holds an 8 character string specifying the date of the firmware release. The date format is **DD/MM/YY**.

The **Logical Position of Import Export Element** field holds a 4 character string specifying the logical position (address) of the import export element. The format of the string is **mmtt** whereby **mm** = magazine address and **tt** = slot address, e.g. ‘0520’ means the import export element is originated at magazine address 5 slot address 20. This field is right bound and holds leading ASCII zeros if necessary.

The **Maximum Magazine Count** field holds a 2 character string specifying the maximum number of magazines the unit is capable of storing, e.g. '03' means the unit is capable of holding up to 3 magazines. Each Magazine is capable of holding a maximum number of media defined by the *Maximum Medium Count* field. The **Maximum Magazine Count** field is right bound and holds leading ASCII zeros if necessary.

The **Maximum Media Count** field holds a 2 character string specifying the maximum number of media a single magazine is capable of storing, e.g. '50' means one magazine may hold up to 50 media. This field is right bound and holds leading ASCII zeros if necessary.

The **Data Transfer Elements Count** field holds a 2 character string specifying the number of data transfer elements equipped in the unit, e.g. '04' means the unit is equipped with 4 data transfer elements. This field is right bound and holds a leading ASCII zero if necessary.

The **Logical Position of Data Transfer Element** field(s) hold(s) a 4 character string (each) specifying the logical position (address) of each data transfer element. The format of the string is **mmtt** whereby **mm** = **magazine address** and **tt** = **slot address**, e.g. '0308' means this data transfer element is originated at magazine address 3 slot address 8. This field is right bound and holds leading ASCII zeros if necessary.

### 3.4 Move Medium

The MOVE MEDIUM command requests the target move a unit of media from a source element to a destination element.

Table 10: MOVE MEDIUM Command

| Bit Byte | 7                         | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|---------------------------|---|---|---|---|---|---|---|
| 0        | ID (hex)                  |   |   |   |   |   |   |   |
| 1        | Operation Code (0x24)     |   |   |   |   |   |   |   |
| 2        | (MSB) Source Address      |   |   |   |   |   |   |   |
| 5        | (LSB)                     |   |   |   |   |   |   |   |
| 6        | (MSB) Destination Address |   |   |   |   |   |   |   |
| 9        | (LSB)                     |   |   |   |   |   |   |   |
| 10       | (MSB) CRC16               |   |   |   |   |   |   |   |
| 13       | (LSB)                     |   |   |   |   |   |   |   |
| 14       | RDB Terminator (0x00)     |   |   |   |   |   |   |   |

The source address specifies the location that the medium is taken from, and the destination address specifies the location that the medium is moved to.

The **Source Address** field holds a 4 character string identifying the address of the storage element that is to be moved. The format of the string is **mmtt** whereby **mm** = **magazine address** and **tt** = **slot address**, e.g. '0108' means magazine address 1 and slot address 8.

The **Destination Address** field holds a 4 character string identifying the address of the storage element where the medium specified by the **Source Address** field has to be moved to. The format of the string is **mmtt** whereby **mm** = **magazine address** and **tt** = **slot address**, e.g. '0350' means magazine address 3 and slot address 50.

If an attempt is made to export or import a medium from or to a logical position that has previously been reserved using the WriteSESW (see *Chapter 3.11 WriteSESW* on page 36) command with a value of **0x08** the command will be terminated and the CES field will be set to UNIT ATTENTION: RESERVATION CONFLICT. No media will be moved in this case.

### 3.5 ReadSESW

The READSESW command requests the **Mercury** to report the Storage Element Status Word (SESW) of either all *storage elements*, a specific magazine or the specific storage element depending on the command parameters.

**NOTE:**

This command does not reflect the current status of the media changer. It reports which storage addresses (slots within magazines) permanently hold a unit of media and which storage positions are free (do not hold a unit of media) as well as which storage element positions are reserved and which storage element positions have been exchanged.

**Table 11: ReadSESW Command**

| Bit<br>Byte | 7                             | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|-------------------------------|---|---|---|---|---|---|---|
| 0           | ID (hex)                      |   |   |   |   |   |   |   |
| 1           | Operation Code (0x27)         |   |   |   |   |   |   |   |
| 2           | (MSB) Storage Element Address |   |   |   |   |   |   |   |
| 5           | (LSB)                         |   |   |   |   |   |   |   |
| 6           | (MSB) CRC16                   |   |   |   |   |   |   |   |
| 9           | (LSB)                         |   |   |   |   |   |   |   |
| 10          | RDB Terminator (0x00)         |   |   |   |   |   |   |   |

The **Storage Element Address** field holds a 4 character string specifying which status should be returned. The format of the string is **mmtt** where **mm** specifies the magazine address and **tt**

specifies the tray address (see *Table 16* for valid values).

**Table 12: Valid Values for Storage Element Address Field**

| mmtt | Description of Parameters  |
|------|--|
| 0000 | Return SESW for all 150 CDs  |
| 0m00 | Return SESW for magazine 0m, where m = 1 -3                          |
| 0mtt | Return SESW for tray tt of magazine 0m, where tt = 01-50 and m = 1-3 |

**Table 13: ReadSESW RDB for mmtt 0000**

| Bit Byte | 7  | 6                        | 5 | 4 | 3   | 2 | 1     | 0 |  |
|----------|--|--------------------------|---|---|-----|---|-------|---|--|
| 0        | ID (hex)                                   |                          |   |   |     |   |       |   |  |
| 1        | Operation Code (0x27)                      |                          |   |   |     |   |       |   |  |
| 2        | (MSB)                                      | Command Execution Status |   |   |     |   |       |   |  |
| 3        |  |                          |   |   |     |   | (LSB) |   |  |
| 4        | (MSB)                                      | Magazine Address         |   |   |     |   |       |   |  |
| 5        |  |                          |   |   |     |   | (LSB) |   |  |
| 6        |  |                          |   |   | '0' |   |       |   |  |
| 7        |  |                          |   |   | '0' |   |       |   |  |
| 8        | (Higher Nibble)Storage Element Status Word |                          |   |   |     |   |       |   |  |
| 9        | (Lower Nibble)                             |                          |   |   |     |   |       |   |  |
| 306      | (Higher Nibble)Storage Element Status Word |                          |   |   |     |   |       |   |  |
| 307      | (Lower Nibble)                             |                          |   |   |     |   |       |   |  |
| 308      | (MSB)                                      | CRC16                    |   |   |     |   |       |   |  |
| 311      |  |                          |   |   |     |   | (LSB) |   |  |
| 312      | RDB Terminator (0x00)                      |                          |   |   |     |   |       |   |  |

If the command requested to return the status word for all CDs the **Magazine Address** field holds a 2 character string with the val-

ue '00' indicating that the SESW for all elements are returned.

**Storage Element Status Word** field holds a 2 character string identifying the status word for this storage element address. The first character holds the higher nibble of the SESW and the second character holds the lower nibble of the SESW. The storage element status field is defined in .

**Table 14: ReadSESW RDB for 0m00**

| Bit Byte | 7  | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|--|---|---|---|---|---|---|---|
| 0        | ID (hex)                                   |   |   |   |   |   |   |   |
| 1        | Operation Code (0x27)                      |   |   |   |   |   |   |   |
| 2        | (MSB) Command Execution Status             |   |   |   |   |   |   |   |
| 3        | (LSB)                                      |   |   |   |   |   |   |   |
| 4        | (MSB) Magazine Address                     |   |   |   |   |   |   |   |
| 5        | (LSB)                                      |   |   |   |   |   |   |   |
| 6        | '0'  |   |   |   |   |   |   |   |
| 7        | '0')                                       |   |   |   |   |   |   |   |
| 8        | (Higher Nibble)Storage Element Status Word |   |   |   |   |   |   |   |
| 9        | (Lower Nibble)                             |   |   |   |   |   |   |   |
| 106      | (Higher Nibble)Storage Element Status Word |   |   |   |   |   |   |   |
| 107      | (Lower Nibble)                             |   |   |   |   |   |   |   |
| 108      | (MSB) CRC16                                |   |   |   |   |   |   |   |
| 111      | (LSB)                                      |   |   |   |   |   |   |   |
| 112      | RDB Terminator (0x00)                      |   |   |   |   |   |   |   |

If the command requested to return the status word for a certain magazine the **Magazine Address** field holds a 2 character string identifying the address of the magazine the element status words are being reported for. The format of the string is mm, e.g. '02' means magazine address 2.

The **Storage Element Status Word** field holds a 2 character string identifying the status word for this storage element address.

The first character holds the higher nibble of the SESW and the second character holds the lower nibble of the SESW. The storage element status field is defined in *Table 16*.

**Table 15: ReadSESW RDB for 0mtt**

| Bit<br>Byte | 7                     | 6                           | 5 | 4 | 3 | 2 | 1 | 0              |  |
|-------------|-----------------------|-----------------------------|---|---|---|---|---|----------------|--|
| 0           | ID (hex)              |                             |   |   |   |   |   |                |  |
| 1           | Operation Code (0x27) |                             |   |   |   |   |   |                |  |
| 2           | (MSB)                 | Command Execution Status    |   |   |   |   |   |                |  |
| 3           |                       |                             |   |   |   |   |   | (LSB)          |  |
| 4           | (MSB)                 | Magazine Address            |   |   |   |   |   |                |  |
| 5           |                       |                             |   |   |   |   |   | (LSB)          |  |
| 6           | (MSB)                 | Tray Address                |   |   |   |   |   |                |  |
| 7           |                       |                             |   |   |   |   |   | (LSB)          |  |
| 8           | (Higher Nibble)       | Storage Element Status Word |   |   |   |   |   |                |  |
| 9           |                       |                             |   |   |   |   |   | (Lower Nibble) |  |
| 10          | (MSB)                 | CRC16                       |   |   |   |   |   |                |  |
| 13          |                       |                             |   |   |   |   |   | (LSB)          |  |
| 14          | RDB Terminator (0x00) |                             |   |   |   |   |   |                |  |

If the command requested to return the status word for a certain tray within a magazine the **Magazine Address** field holds a 2 character string identifying the magazine address the element status word is reported for. The format of the string is **mm**, e.g. **'02'** means magazine address **2**.

The **Tray Address** field holds a 2 character string identifying the tray for which the status word is being reported. The format of the string is **mm**, e.g. **'01'** means magazine address **1**.

The **Storage Element Status Word** field holds a 2 character string identifying the status word for this storage element address. The first character holds the higher nibble of the SESW and the second character holds the lower nibble of the SESW. The storage element status field is defined in .

**Table 16: Storage Element Status Word**

| Bit | Name                  | Set by                             | Reset by  | Nibble |
|-----|-----------------------|------------------------------------|---|--------|
| 0   | <b>CDDetected</b>     | <b>Mercury when CD is detected</b> | <b>Mercury when no CD is detected</b>               | lower  |
| 1   | <b>CDDValid</b>       | <b>CD detection</b>                | <b>opening door or Move command to Mail-slot</b>    | lower  |
| 2   | <b>CDValid</b>        | <b>WriteSESW</b>                   | <b>WriteSESW, closing-opening door or Mail-slot</b> | lower  |
| 3   | <b>CDExpPermitted</b> | <b>WriteSESW</b>                   | <b>WriteSESW, closing-opening door or Mail-slot</b> | lower  |
| 4   | <b>CDChanged</b>      | <b>closing Mail-slot</b>           | <b>WriteSESW or opening door</b>                    | higher |
| 5   | <b>Reserved</b>       |                                    |   |        |
| 6   | <b>Reserved</b>       |                                    |   |        |
| 7   | <b>UserDefined</b>    | <b>WriteSESW</b>                   | <b>WriteSESW, closing-opening door or Mail-slot</b> | higher |

The **CDDetected (CDD)** bit is tightly linked to the **CDDValid** bit and is **undefined** as long as the **CDDValid** bit equals **0**.

If the **CDDetected (CDD)** bit equals **1** and the **CDDValid** bit equals **1** it identifies that a CD is mounted to this particular tray. This bit will be set by the **Mercury** whenever the **Mercury** moves a CD from its home position to any other location (e.g. when proceeding a Move, Exchange or Initialize Element Status command) and detects a CD in that tray.

The **CDDetected (CDD)** bit will be reset to **0** by the **Mercury**

whenever the **Mercury** moves a CD from its home position to any other location (e.g. when proceeding a Move, Exchange or Initialize Element Status command) and does not detect a CD in that tray.

If the **CDDValid** bit equals **1** it identifies that a detection for the CDD bit of this tray has been performed due to a move of this tray from its home position to any other location (e.g. when proceeding a Move, Exchange or Initialize Element Status command). A value of **1** for the **CDDValid** bit indicates that the **CDDetect** bit is valid now. This **CDDValid** bit will be reset to **0** for all trays whenever the doors of the **Mercury** are opened or reset to **0** for a specific tray whenever this tray is being moved to the Mail-slot.

The **CDValid** bit identifies if a CD is valid (e.g. if it is a ISO9660 CD). This bit is user definable and has to be set using the WriteSESW command. The **CDValid** bit may also be reset using the WriteSESW command. The **CDValid** bit will be reset whenever the doors of the **Mercury** are closed.

Setting the **CDExpPermitted** bit to **1** identifies that a CD may be exported through the Mail-slot. Setting the **CDExpPermitted** bit to **0** reserves a CD. A reserved CD may not be exported through the Mail-slot. This bit may be set using the WriteSESW command and may be reset using the WriteSESW command. The **CDExpPermitted** bit will be reset (set to 0) whenever the doors of the **Mercury** are closed.

The **CDChanged** bit indicates that a CD has been changed (imported or exported through the Mail-slot). This bit will be set whenever the Mail-slot has been closed. The **CDChanged** bit may be reset using the WriteSESW command.

The **UserDefined** bit may be used for different purposes. This bit may be set or reset using the WriteSESW command. The **UserDefined** bit will be reset whenever the doors of the **Mercury** are closed.

### 3.6 Request Device ID

The REQUEST DEVICE ID command is used to retrieve information about the drive IDs of the drives mounted in the **Mercury**.

**Table 17: REQUEST DEVICE ID Command**

| Bit<br>Byte | 7                     | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|-----------------------|---|---|---|---|---|---|---|
| 0           | ID (hex)              |   |   |   |   |   |   |   |
| 1           | Operation Code (0x2C) |   |   |   |   |   |   |   |
| 2           | CRC16                 |   |   |   |   |   |   |   |
| 5           |                       |   |   |   |   |   |   |   |
| 6           | RDB Terminator (0x00) |   |   |   |   |   |   |   |

Table 18: REQUEST DEVICE ID RDB

| Bit Byte | 7                     | 6                        | 5 | 4 | 3 | 2 | 1     | 0     |
|----------|-----------------------|--------------------------|---|---|---|---|-------|-------|
| 0        | ID (hex)              |                          |   |   |   |   |       |       |
| 1        | Operation Code (0x2C) |                          |   |   |   |   |       |       |
| 2        | (MSB)                 | Command Execution Status |   |   |   |   |       |       |
| 3        |                       |                          |   |   |   |   |       | (LSB) |
| 4        | (MSB)                 | Drive1 ID                |   |   |   |   |       |       |
| 5        |                       |                          |   |   |   |   | (LSB) |       |
| 6        | (MSB)                 | Drive2 ID                |   |   |   |   |       |       |
| 7        |                       |                          |   |   |   |   | (LSB) |       |
| 8        | (MSB)                 | Drive3 ID                |   |   |   |   |       |       |
| 9        |                       |                          |   |   |   |   | (LSB) |       |
| 10       | (MSB)                 | Drive4 ID                |   |   |   |   |       |       |
| 11       |                       |                          |   |   |   |   | (LSB) |       |
| 12       | (MSB)                 | Reserved                 |   |   |   |   |       |       |
| 13       |                       |                          |   |   |   |   | (LSB) |       |
| 14       | (MSB)                 | CRC16                    |   |   |   |   |       |       |
| 17       |                       |                          |   |   |   |   | (LSB) |       |
| 18       | RDB Terminator (0x00) |                          |   |   |   |   |       |       |

The **Drive1 ID** field holds a 2 character string identifying the SCSI ID assigned to the drive with logical address '0001'.

The **Drive2 ID** field holds a 2 character string identifying the SCSI ID assigned to the drive with logical address '0002'.

The **Drive3 ID** field holds a 2 character string identifying the SCSI ID assigned to the drive with logical address '0003'.

The **Drive4 ID** field holds a 2 character string identifying the SCSI ID assigned to the drive with logical address '0004'.

If any of the drives is not mounted to the **Mercury** the field will hold '--' characters

### 3.7 Request Exchange Status

The REQUEST EXCHANGE STATUS command (*Table 17*) instructs the **Mercury** to return the status of an EXCHANGE MEDIUM command. The REQUEST EXCHANGE STATUS command should be used by applications to ensure that a medium exchange has been performed properly (e.g. to see if the command has been proceeded by the **Mercury** despite a power drop or failure). The general use of this command in conjunction with the EXCHANGE MEDIUM command is shown in *Chapter 3.1 Exchange Medium* on page 2.

Table 19: REQUEST EXCHANGE STATUS command

| Bit Byte | 7                     | 6     | 5 | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|-------|---|---|---|---|---|-------|--|
| 0        | ID (hex)              |       |   |   |   |   |   |       |  |
| 1        | Operation Code (0x2B) |       |   |   |   |   |   |       |  |
| 2        | (MSB)                 | CRC16 |   |   |   |   |   |       |  |
| 5        |                       |       |   |   |   |   |   | (LSB) |  |
| 6        | RDB Terminator (0x00) |       |   |   |   |   |   |       |  |

Table 20: REQUEST EXCHANGE STATUS RDB

| Bit Byte | 7                     | 6                        | 5 | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|--------------------------|---|---|---|---|---|-------|--|
| 0        | ID (hex)              |                          |   |   |   |   |   |       |  |
| 1        | Operation Code (0x2B) |                          |   |   |   |   |   |       |  |
| 2        | (MSB)                 | Command Execution Status |   |   |   |   |   |       |  |
| 3        |                       |                          |   |   |   |   |   | (LSB) |  |
| 4        | (MSB)                 | MSW                      |   |   |   |   |   |       |  |
| 5        |                       |                          |   |   |   |   |   | (LSB) |  |
| 6        | Door Status           |                          |   |   |   |   |   |       |  |
| 7        | Keylock Status        |                          |   |   |   |   |   |       |  |
| 8        | (MSB)                 | Exchange ID              |   |   |   |   |   |       |  |
| 11       |                       |                          |   |   |   |   |   | (LSB) |  |

Table 20: REQUEST EXCHANGE STATUS RDB

|    |       |                              |       |
|----|-------|------------------------------|-------|
| 12 | (MSB) | Previous Exchange Parameters |       |
| 19 |       |                              | (LSB) |
| 20 | (MSB) | Current Exchange Parameters  |       |
| 27 |       |                              | (LSB) |
| 28 | (MSB) | Exchange Status              |       |
| 29 |       |                              | (LSB) |
| 30 | (MSB) | CRC16                        |       |
| 33 |       |                              | (LSB) |
| 34 |       | RDB Terminator (0x00)        |       |

For an explanation of the **Machine Status Word (MSW)** see *Chapter 3.11 WriteSESW* on page 36.

For an explanation of the **Door Status** see *Chapter 3.8 Request Status* on page 23.

For an explanation of the **Keylock Status** see *Chapter 3.8 Request Status* on page 23.

The **Exchange ID** field holds a unique 4 character string identifying the ID used by the **Mercury** to identify the exchange process. The range of the ID is from '0001' to '9999'.

If the changer returns all zero ('0000'), no change has been proceeded up to then.

The **Previous Exchange Parameters** field holds an 8 character string identifying the addresses of the storage elements of the last successful exchange command.

The format of the string is **mmtt1mmtt2**, where  
**mmtt1 = address of first storage element,**  
**mmtt2 = address of second storage element,**  
 e.g. '01080344' means that the first storage element in magazine 1 slot 8 has successfully been exchanged with the second storage element at magazine 3 slot 44.

The **Current Exchange Parameters** field holds an 8 character string identifying the addresses of the storage elements that will be exchanged.

The format of the string is **mmtt1mmtt2**, where  
**mmtt1 = address of first storage element,**  
**mmtt2 = address of second storage element,**  
 e.g. **'01080344'** means that the first storage element in magazine 1 slot 8 will be exchanged with the second storage element at magazine 3 slot 44.

The **Exchange Status** field holds a 2 character string identifying the status of the exchange command (see *Table 21* for possible values).

**Table 21: Exchange Status Values**

| CES-Code | Description                                     |
|----------|---|
| 0x       | Ready Group                                     |
| 0x00     | Exchange successfully completed                 |
| 1x       | Busy Group                                      |
| 0x10     | Exchange in progress                            |
| 3x       | Failure Group                                   |
| 0x30     | Fatal failure while proceeding exchange command |

#### 3.7.1 Request Exchange Status Command in Conjunction with the Exchange Medium Command

The following procedure describes how to make use of both commands and how to determine if an exchange process has been completed successfully even if during command proceeding there was a loss of power.

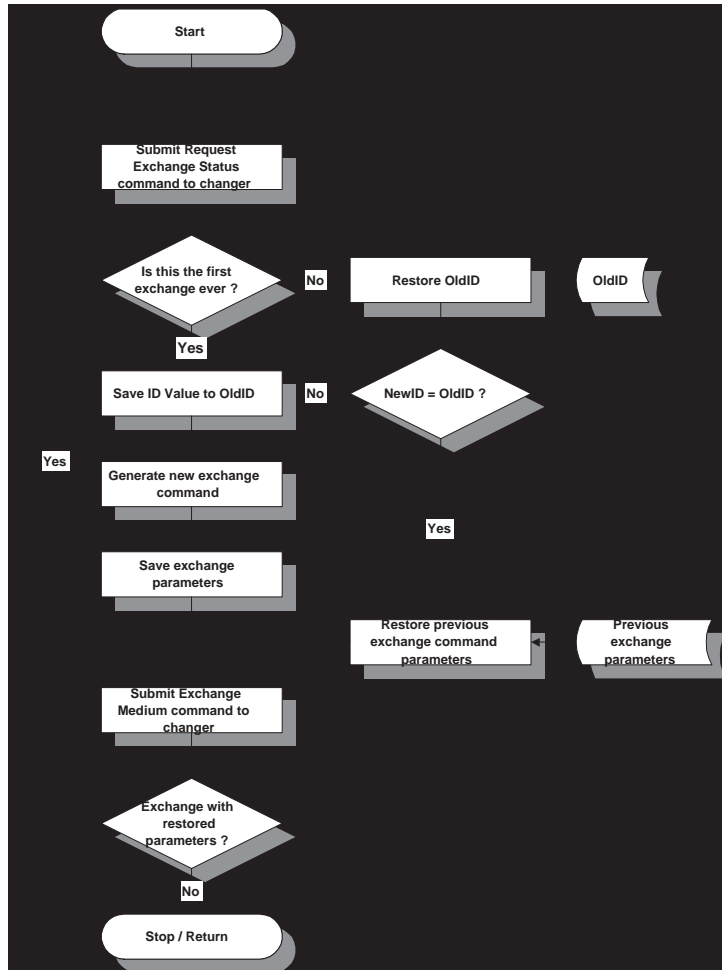
- 1) **Request an exchange ID using the REQUEST EXCHANGE STATUS command.**
- 2) **Test if the ID returned is zero.**
  - a) If it is zero (this indicates that this is the very first EXCHANGE MEDIUM command)
    - Store the ID in a non-volatile memory location of the workstation
    - Store the parameters of the exchange in a non-volatile memory location of the workstation
    - Transmit the desired exchange command to the **Mercury**
    - Continue with step 4
  - b) If it is non zero  
load the previously stored ID from the non-volatile memory location of the workstation
- 3) **Test if the new ID equals the old ID**
  - a) They are not equal:
    - Store the parameters of the exchange in a non-volatile memory location of the workstation
    - Transmit the EXCHANGE MEDIUM command to the **Mercury**
    - Continue with step 4
  - b) They are equal (the previous change has not been proceeded):
    - Restore the previously saved parameters
    - Transmit the restored EXCHANGE MEDIUM command to the **Mercury**.
    - Continue with step 4

- 4) **Request the exchange status using the REQUEST EXCHANGE STATUS command.**
- 5) **Test if the command has completed (Status = '00')**
  - a) It has completed:
    - Continue with step 6
  - b) It has not completed:
    - Continue with step 4
  - c) It has completed with error:
    - Continue with step error recovery
- 6) **Test if it was an EXCHANGE MEDIUM command with restored parameters**
  - a) Exchange with restored parameters:
    - Continue with step 1
  - b) Exchange with current parameters:
    - **Done**

## Operation Codes

### Request Exchange Status

Figure 1: How to use the REQUEST EXCHANGE STATUS command



### 3.8 Request Status

The REQUEST STATUS command is used to get information about the state the **Mercury** is currently in. This information includes the following:

- **Machine Status Word (MSW)**
- status of the **Mercury Doors** (open/closed)
- status of the **Mercury Keylock** (locked/unlocked)
- status of the **Mercury Mail-slot** (opening, closing, open, closed)
- status of each **Data Transfer Element** (disc loaded, loading, no disc loaded, ejecting)

Table 22: REQUEST STATUS Command

| Bit Byte | 7                     | 6 | 5     | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|---|-------|---|---|---|---|-------|--|
| 0        | ID (hex)              |   |       |   |   |   |   |       |  |
| 1        | Operation Code (0x22) |   |       |   |   |   |   |       |  |
| 2        | (MSB)                 |   | CRC16 |   |   |   |   |       |  |
| 5        |                       |   |       |   |   |   |   | (LSB) |  |
| 6        | RDB Terminator (0x00) |   |       |   |   |   |   |       |  |

Table 23: REQUEST STATUS RDB

| Bit Byte | 7                     | 6                         | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|-----------------------|---------------------------|---|---|---|---|---|---|
| 0        | ID (hex)              |                           |   |   |   |   |   |   |
| 1        | Operation Code (0x22) |                           |   |   |   |   |   |   |
| 2        | (MSB)                 | Command Execution Status  |   |   |   |   |   |   |
| 3        | (LSB)                 |                           |   |   |   |   |   |   |
| 4        | (MSB)                 | Machine Status Word (MSW) |   |   |   |   |   |   |
| 5        | (LSB)                 |                           |   |   |   |   |   |   |
| 6        | Door Status           |                           |   |   |   |   |   |   |
| 7        | Keylock Status        |                           |   |   |   |   |   |   |
| 8        | (MSB)                 | Mail-slot Status          |   |   |   |   |   |   |
| 15       | (LSB)                 |                           |   |   |   |   |   |   |
| 16       | (MSB)                 | Drive (1) Status          |   |   |   |   |   |   |
| 23       | (LSB)                 |                           |   |   |   |   |   |   |
|          | Drive N Status        |                           |   |   |   |   |   |   |
| 24+(N*8) | (MSB)                 | CRC16                     |   |   |   |   |   |   |
| 27+(N*8) | (LSB)                 |                           |   |   |   |   |   |   |
| 28+(N*8) | RDB Terminator (0x00) |                           |   |   |   |   |   |   |

The **Machine Status Word** field holds a two character string specifying the current status of the **Mercury** device, see *Table 24*.

Table 24: Machine Status Word

| MSW-Code  | MSW Description                         |
|-----------|---|
| <b>0x</b> | <b>Ready Group</b>                      |
| 00        | machine is ready to process any command |
| <b>2x</b> | <b>Busy Group</b>                       |
| 20        | Power On Self Test in progress (POST)   |
| 21        | Diagnostics procedure in progress       |
| 22        | Rezero Unit in progress                 |
| 24        | Initialize Element Status in progress   |
| 25        | Manual administration in progress       |
| <b>3x</b> | <b>Failure Group</b>                    |
| 30        | General robotics Failure                |

**NOTE:**

Whenever the MSW equals '00' the machine is ready to proceed any command.

If the MSW holds a value of the **BUSY** group the machine is not ready to proceed certain commands. The machine typically will become ready within a short time (<15s).

If the MSW holds a value of the **HARDWARE FAILURE** group the machine is not operational any more. In this specific case the machine will only accept a **REQUEST STATUS** command and a **REZERO UNIT** command. The **Door Status** field holds an ASCII field specifying the door status of the media changer device. A value of '1' indicates that the door of the **Mercury** is open. A value of '0' indicates that the door of the **Mercury** is closed.

The **Keylock Status** field holds an ASCII field specifying the Keylock status of the media changer device. A value of '01' indicates that the Keylock is locked. A value of '10' indicates that the Keylock is unlocked.

The **Mail-slot Status** field holds an 8 character string specifying the source address of the medium mounted to the import export element and the import export element status.

The format of the string is **mmttlms** where

**mm** = magazine address,

**tt** = slot address,

**ls** = Mail-slot loader status, and

**ms** = Mail-slot status;

e.g. '0221xxzz' means the import export element has mounted the medium originated in magazine 2, slot 21, for status code information see *Table 25* and *Table 26*.

The **Drive Status** field holds an 8 character string specifying the source address of the medium mounted to the data transfer element and the data transfer element status. The format of the string is **mmttlnds** whereby **mm** = magazine address, **tt** = slot address, **ls** = drive loader status and **ds** = drive status, e.g. '0333xxzz' means the data transfer element has mounted the medium originated in magazine 03, slot 33, for status code information see *Table 27* and *Table 28*.

- **Group Codes 0x**

The specific element is ready to perform an action.

- **Group Codes 1x**

The specific element is currently busy.

- **Group Codes 3x**

The specific element run into a failure that could be recovered.  
The element is operational.

- **Group Codes 4x**

The specific element run into a fatal failure. The element is not operational any more.

**Table 25: Mail-slot Loader Status Codes**

**Group '0' Codes**

'00' tray loaded

'01' loader empty

**Group '1' Codes**

'10' loading tray

'11' unloading tray

**Group '2' Codes**

'21' Tray home position changed

**Group '3' Codes**

'30' error loading tray

'31' error unloading tray

**Table 26: Mail-slot Status Codes**

**Group '0' Codes**

'00' Mail-slot open

'01' Mail-slot closed

**Group '1' Codes**

'10' Mail-slot opening

'11' Mail-slot closing

**Group '2' Codes**

'21' Mail-slot manually closed

**Group '3' Codes**

'30' error opening Mail-slot

'31' error closing Mail-slot

**Group '4' Codes**

'40' fatal Mail-slot failure (Mail-slot stuck)

**Table 27: Drive Loader Status Codes**

**Group ‘0’ Codes**

‘00’ tray loaded

‘01’ loader empty

**Group ‘1’ Codes**

‘10’ loading tray

‘11’ unloading tray

**Group ‘2’ Codes**

‘21’ Tray home position changed

**Group ‘3’ Codes**

‘30’ error loading tray

‘31’ error unloading tray

**Table 28: Drive Status Codes**

**Group ‘0’ Codes**

‘00’ loader down

‘01’ loader up

**Group ‘1’ Codes**

‘10’ lowering loader

‘11’ ejecting loader

**Group ‘2’ Codes**

‘21’ Loader auto-ejected

**Group ‘3’ Codes**

‘30’ error lowering loader

‘31’ error ejecting loader

**Group ‘4’ Codes**

‘40’ fatal loader failure (loader stuck)

### 3.9 Rezero Unit

The REZERO UNIT command is used to drive the **Mercury** into a pre-defined state. The desired state is defined by the **Action Code** parameter

Table 29: REZERO UNIT Command

| Bit<br>Byte | 7                     | 6           | 5 | 4 | 3 | 2 | 1 | 0     |  |
|-------------|-----------------------|-------------|---|---|---|---|---|-------|--|
| 0           | ID (hex)              |             |   |   |   |   |   |       |  |
| 1           | Operation Code (0x20) |             |   |   |   |   |   |       |  |
| 2           | (MSB)                 | Action Code |   |   |   |   |   |       |  |
| 3           |                       |             |   |   |   |   |   | (LSB) |  |
| 4           | (MSB)                 | CRC16       |   |   |   |   |   |       |  |
| 7           |                       |             |   |   |   |   |   | (LSB) |  |
| 8           | RDB Terminator (0x00) |             |   |   |   |   |   |       |  |

The **Action Code** parameter is a two character string that specifies which action is to be performed by the REZERO UNIT command. shows a list of possible action codes and the action they will perform.

**Table 30: Action Codes for REZERO UNIT Command**

**Action Code ‘00’:**

- reset hardware failure condition (MSW=‘3x’, CES=‘30’)

**Action Code ‘01’:**

- reset hardware failure condition (MSW=‘3x’, CES=‘30’)
- the *data transfer element(s)* is/are unmounted
- *media* are returned to their storage locations
- the *import export element* is driven into its closed position

**Action Code ‘02’:**

- reset hardware failure condition (MSW=‘3x’, CES=‘30’)
- the *data transfer element(s)* is/are unmounted
- *media* are returned to their storage locations
- the *import export element* is driven into its closed position
- the *medium transport element* is driven to its home position
- the door of the **Mercury** is opened
- 

## 3.10 UI Command

Display and key-pad are used from several independent processes within the **Mercury** firmware. Each process owns a virtual display that is available for output at any time. The virtual display is activated by the process. At the same time the key-pad is connected to the process and the key-pad buffer is erased.

The UI command allows to redirect display and key-pad to the host interface (RS-232 connection).

ATTENTION:

THE HOST SYSTEM HAS TO RELEASE THE DISPLAY (OPTION 03) OTHERWISE THERE IS NO POSSIBILITY FOR THE **Mercury** TO ACCESS THE DISPLAY.

**Table 31: UI Command**

| Bit Byte | 7                       | 6           | 5 | 4 | 3 | 2 | 1     | 0 |  |
|----------|-------------------------|-------------|---|---|---|---|-------|---|--|
| 0        | ID (hex)                |             |   |   |   |   |       |   |  |
| 1        | Operation Code (0x30)   |             |   |   |   |   |       |   |  |
| 2        | (MSB)                   | Action Code |   |   |   |   |       |   |  |
| 3        |                         |             |   |   |   |   | (LSB) |   |  |
| 4        | N Bytes Data (optional) |             |   |   |   |   |       |   |  |
| 4+N      | (MSB)                   | CRC16       |   |   |   |   |       |   |  |
| 7+N      |                         |             |   |   |   |   | (LSB) |   |  |
| 8+N      | RDB Terminator (0x00)   |             |   |   |   |   |       |   |  |

The **Action Code** parameter is a two character string that specifies which action is to be performed by the UI command. *Table 32* shows a list of possible action codes and the action they will perform

The **Data Bytes** of the UI command are optional and are only used for the WRITE TEXT action parameter. For all the other action parameters there are no data bytes to provide.

### 3.10.1 UI Action Parameter Description.

Following is a detailed description of the various parameters to the UI command.

**Table 32: UI Action Codes**

| Option | Description              |
|--------|--------------------------|
| 00     | UI status inquiry        |
| 01     | Activate Virtual Display |
| 02     | Release Virtual Display  |
| 03     | Write Text               |
| 04     | Read Key-Pad Buffer      |

### Status Inquiry

The STATUS INQUIRY option is used to determine whether the UI is active (re-directed) or is not active (not re-directed).

Table 33: UI Status Inquiry RDB

| Bit Byte | 7                     | 6         | 5 | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|-----------|---|---|---|---|---|-------|--|
| 0        | ID (hex)              |           |   |   |   |   |   |       |  |
| 1        | Operation Code (0x30) |           |   |   |   |   |   |       |  |
| 2        | 0                     |           |   |   |   |   |   |       |  |
| 3        | 0                     |           |   |   |   |   |   |       |  |
| 4        | (MSB)                 | UI Status |   |   |   |   |   |       |  |
| 5        |                       |           |   |   |   |   |   | (LSB) |  |
| 6        | (MSB)                 | CRC16     |   |   |   |   |   |       |  |
| 9        |                       |           |   |   |   |   |   | (LSB) |  |
| 10       | RDB Terminator (0x00) |           |   |   |   |   |   |       |  |

The **UI Status field** is a 2 character field that identifies the status of the display and key-pad. A value of **'00'** indicates that the **UI is not active**. A value of **'01'** indicates that the **UI is active**.

### Activate Virtual Display

The ACTIVATE VIRTUAL DISPLAY option is used to activate (re-directed) the virtual display and key-pad.

### Deactivate virtual Display

The DEACTIVATE VIRTUAL DISPLAY option is used to deactivate the virtual display and key-pad.



### Write Text

The Write Text action parameter is used to display text on the display of the **Mercury** jukebox. The text string has to be provided in the UI commands optional data fields (see **Table 31**). Since the display of the **Mercury** is a **16 \* 2 LC-display**, the maximum allowable string length (excluding control characters) is 16 characters.

The output string (characters) may be formatted in a **'printf'** like manner. Possible characters are all visible characters from **20h to 7Fh**. Additionally the following control characters may be used

**Table 34: Control Characters**

| ANSI-C | Function  |
|--------|---|
| "\f"   | Clear Display   |
| "\t"   | activate character blinking for all characters that follow                |
| "\v"   | deactivate character blinking for all characters that follow              |
| "\b"   | Back Space  |
| "\r"   | Carriage Return   |
| "\n"   | New Line  |
| "\a"   | Buzzer (Beep)   |
| "\x1e" | Position cursor to top left corner, display contents will not be affected |

### Read Key-pad

The READ KEY-PAD action parameter is used to retrieve key strokes from the virtual key-pad of the **Mercury**.

Table 35: UI Read Key-Pad RDB

| Bit Byte | 7                     | 6     | 5 | 4 | 3 | 2 | 1 | 0     |  |
|----------|-----------------------|-------|---|---|---|---|---|-------|--|
| 0        | ID (hex)              |       |   |   |   |   |   |       |  |
| 1        | Operation Code (0x30) |       |   |   |   |   |   |       |  |
| 2        | 0                     |       |   |   |   |   |   |       |  |
| 3        | 1                     |       |   |   |   |   |   |       |  |
| 4        | 0                     |       |   |   |   |   |   |       |  |
| 5        | n                     |       |   |   |   |   |   |       |  |
| 6        | String (of length L)  |       |   |   |   |   |   |       |  |
| 6+L      | (MSB)                 | CRC16 |   |   |   |   |   |       |  |
| 9+L      |                       |       |   |   |   |   |   | (LSB) |  |
| 10+L     | RDB Terminator (0x00) |       |   |   |   |   |   |       |  |

$n \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$  (number of keystrokes) and  
 $0 \leq \text{sizeof}(\text{String}) \leq 16$

Each key stroke (for one or more keys activated at the same time) is translated into an ASCII-String. A maximum of **8** keystrokes will be buffered in a 2 dimensional array (`key_buf [16][8]`). The number of keystrokes (**n**) includes the character actually received by the command.

With each UI:READ KEY-PAD command one buffer element (out of 8; `key_buf [][][n]`) is transmitted. It is possible to activate **0 to 16** keys at the same time (`key_buf [0..15][[]]`). The length of the string is accordingly. Each key relates to an ANSI-C character.

If the buffer is empty (does not hold any keystrokes) the actual key-pad status is transmitted and the number of buffer entries is equal zero ( $n=0$ ).

If there is no keystroke the string (`key_buf [][][n]`) will be **empty** ("").

**Example**

The keystrokes F+1 generate 5 buffer entries (strings).

**Table 36: Key-Pad Buffer Entry Example**

| Time | Key  | Action   | Buffer Contents<br>key_buf [][0..4] | Buffer<br>Entry<br>(n) |
|------|------|--|-------------------------------------|------------------------|
| t1   | none | none   | " "                                 |                        |
| t2   | F    | F key is pressed                                       | " "+ "F"                            |                        |
| t3   | F+1  | F key is held and additionally<br>the 1 key is pressed | " "+ "F" + "F1"                     |                        |
| t4   | F    | 1 key is being released while F<br>key stays pressed   | " "+ "F" + "F1" + "F"               |                        |
| t5   | none | F key is being released                                | " "+ "F" + "F1" + "F" + ""          |                        |
|      |      | key_buf [ ]  | [0][1] [2] [3] [4]                  |                        |

### 3.11 WriteSESW

The WRITESESW command is used to write the Storage Element Status Word (see also *Chapter 3.11 WriteSESW* on page 36).

Table 37: WRITESESW Command

| Bit<br>Byte | 7                     | 6               | 5 | 4 | 3 | 2 | 1 | 0     |  |
|-------------|-----------------------|-----------------|---|---|---|---|---|-------|--|
| 0           | ID (hex)              |                 |   |   |   |   |   |       |  |
| 1           | Operation Code (0x2A) |                 |   |   |   |   |   |       |  |
| 2           | (MSB)                 | Element Address |   |   |   |   |   |       |  |
| 5           |                       |                 |   |   |   |   |   | (LSB) |  |
| 6           | (MSB)                 | SESW Mask       |   |   |   |   |   |       |  |
| 7           |                       |                 |   |   |   |   |   | (LSB) |  |
| 8           | (MSB)                 | CRC16           |   |   |   |   |   |       |  |
| 11          |                       |                 |   |   |   |   |   | (LSB) |  |
| 12          | RDB Terminator (0x00) |                 |   |   |   |   |   |       |  |

The **Element Address** field holds a 4 character string identifying the address of the storage element to be modified. The format of the string is **mmtt** whereby **mm** = **magazine address** and **tt** = **slot address**, e.g. '0108' means magazine address 1 and slot address 8.

The **SESW Mask** field holds a two character string identifying the set/reset mask of the bits to be written to the corresponding Storage Element Status Word.

To set a bit in the SESW Mask the position representing this bit has to be set to 1 (logical one). To clear a bit in the SESW the position representing this bit has to be set to 0 (logical zero). To be shure not to set or reset any bits beside those that you want to change, first read the SESW using the ReadSESW command and then OR or AND the SESW with your SET or RESET mask. Bit positions marked with X will not be affected by a WriteSESW command.

The following bits may be set:

```

Bit No. 7 6 5 4 3 2 1 0
        1 X X X 1 1 X X
    
```

The following bits may be cleared:

```

Bit No. 7 6 5 4 3 2 1 0
        0 X X 0 0 0 X X
    
```

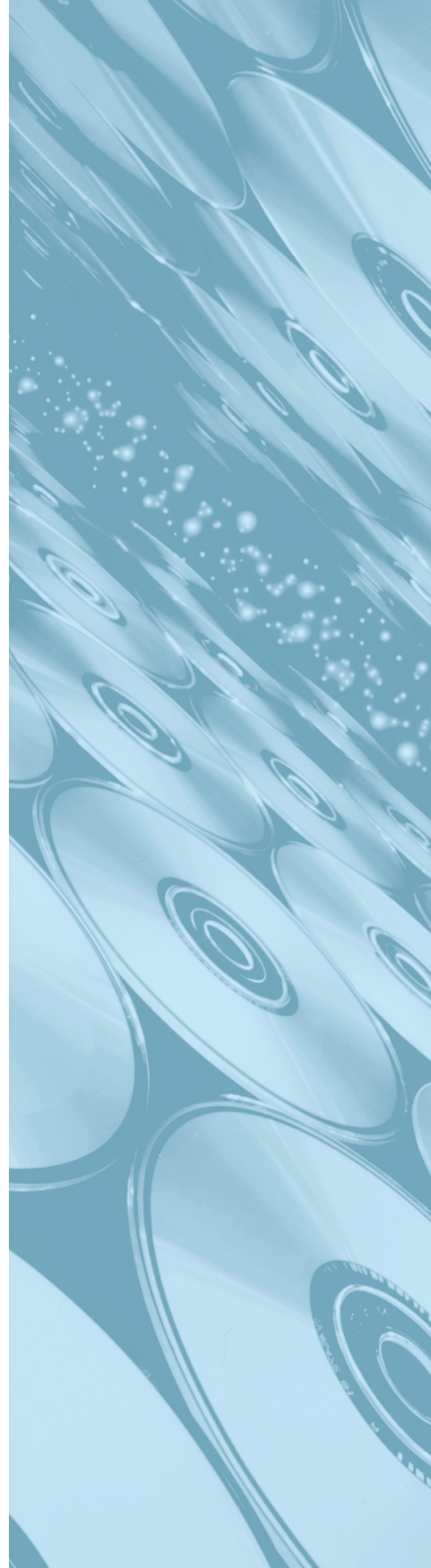
**Table 38: Storage Element Status Word**

| Bit | Name           | Set by                      | Reset by                                     | Nibble |
|-----|----------------|-----------------------------|--|--------|
| 0   | CDDetected     | Mercury when CD is detected | Mercury when no CD is detected               | lower  |
| 1   | CDDValid       | CD detection                | opening door or Move command to Mail-slot    | lower  |
| 2   | CDValid        | WriteSESW                   | WriteSESW, closing-opening door or Mail-slot | lower  |
| 3   | CDExpPermitted | WriteSESW                   | WriteSESW, closing-opening door or Mail-slot | lower  |
| 4   | CDChanged      | closing Mail-slot           | WriteSESW or opening-closing door            | higher |
| 5   | Reserved       |                             |  |        |
| 6   | Reserved       |                             |  |        |
| 7   | UserDefined    | WriteSESW                   | WriteSESW, closing-opening door or Mail-slot | higher |

For an explanation of the SESW bits, please refer to *Chapter 3.11 WriteSESW* on page 36.



# Index





# Index

## A

### Additional Information II

#### addressing

- base address **1-7**
- building block **1-7**
- current element location **1-8**
- drives **1-7**
- element address **1-7**
- magazines **1-8**
- mail-slot **1-7**
- physical to logical **1-7**
- sections **1-7**
- trays **1-8**

## B

### Busy **3-25**

## C

### CES **1-6**

### Changes I

### Characteristics **1-2**

### Command Execution Status **1-6**

### Command processor thread **1-3**

### command queue **1-2**

### Commands

#### Exchange Medium **3-2**

#### Initialize Element Status **3-3**

#### Inquiry **3-4**

#### Move Medium **3-8**

#### ReadSESW **3-9**

#### Request Device ID **3-15**

#### Request Exchange Status **3-17**

#### Request Status **3-23**

#### Rezero Unit **3-29**

#### UI **3-30**

#### WriteSESW **3-36**

### commands

#### acknowledgement **1-4**

#### overlapping **1-4**

#### re-issuing **1-6**

#### Rezero Unit **1-6**

#### validation **1-3**

## D

### Data Transfer Element **3-23**

#### Description

##### Equipment **1-1**

##### Security **1-1**

### Disclaimer I

#### display

##### character set **3-33**

##### control characters **3-33**

##### formatting strings **3-33**

##### output characters **3-33**

##### printf **3-33**

##### retrieve key strokes **3-33**

### Door Status **3-25**

### drive information **3-15**

### Drive Status **3-26**

## E

### elements

#### assigning a status to **1-3**

#### updating status of **1-3**

### Error Recovery **1-6**

### event variable **1-4, 1-5**

## F

### failure condition

#### clearing & unloading drives **1-6**

### failure conditions

#### clearing **1-6**

### fatal failure **1-6**

### First Element Address **3-2**

## H

### Hardware Failure **3-25**

## I

### idle state **1-4**

### Inquiry data

- Data Transfer Elements Count **3-7**
- Logical Position of Data Transfer Element **3-7**
- Logical Position of Import Export Element **3-6**
- Maximum Magazine Count **3-7**
- Maximum Media Count **3-7**
- Part Number **3-6**
- Product Revision Level **3-6**
- Production Date **3-6**
- Robotics Controller Firmware Date **3-6**
- Robotics Controller Firmware Release **3-6**
- Serial Number **3-6**
- Vendor Identification **3-6**
- INVALID COMMAND PARAMETER VALUE **3-3**
- K**
  - Key-Lock **1-2**
  - Keylock Status **3-25**
  - Key-Pad **1-2**
  - key-pad **3-30**
  - Key-Pad Buffer Entry Example **3-35**
- L**
  - Logical Addressing Scheme **1-7**
- M**
  - Machine Status Word **3-23**
  - Mail-slot Status **3-25**
  - Mercury Doors **3-23**
  - Mercury Keylock **3-23**
  - Mercury Mail-slot **3-23**
  - Move Medium
    - Destination Address **3-8**
    - RESERVATION CONFLICT **3-9**
    - Source Address **3-8**
  - Move thread **1-3**
- N**
  - non-recoverable failure **1-6**
  - Notice **I**
- O**
  - Open-Door button **1-2**
  - operating system **1-2**
  - Operation
    - Requirements **1-1**
  - Operator Door Open Request **2-6**
- P**
  - Product Identification **3-6**
- R**
  - RDB **1-6**
  - ReadSESW
    - Magazine Address **3-10**
    - Storage Element Address **3-9**
    - Storage Element Status Word **3-11, 3-13**
  - real-time **1-2**
  - redirect display **3-30**
  - reliability **1-2**
  - Request Status
    - MSW **3-24, 3-25**
  - Response Data Block **1-4**
  - Rezero Unit
    - Action Code **3-29, 3-31**
- S**
  - Second Element Address **3-2**
  - Security
    - key-lock **1-2**
    - software controlled **1-2**
  - SESW
    - bit mask **3-37**
    - CDDetected **3-13**
    - CDDValid **3-13**
    - UserDefined **3-14**
  - status
    - assigned **1-4**
    - element **1-4**
    - event variable **1-4**
    - retrieving **1-5**
  - Status update thread **1-3**

storage elements  
scan for **3-3**

**T**

Termination thread **1-3**  
thread **1-2, 1-3**  
Trademarks **II**

**U**

UAC  
see Unit Attention **1-6**

**UI**

Activate Virtual Display **3-32**  
active **3-32**  
Data Bytes **3-31**  
Deactivate Virtual Display **3-32**  
not active **3-32**

Read Key-pad **3-33**  
Status Inquiry **3-32**  
UI Status field **3-32**  
Unit Attention **1-6**  
command rejected **1-6**  
several conditions pending **1-6**

**V**

virtual display **3-30**  
vital product data **3-4**

**W**

WriteSESW  
clear bit **3-36**  
Element Address **3-36**  
SESW Mask **3-36**  
set bit **3-36**



**Germany:**

**NSM Jukebox GmbH**  
**Im Tiergarten 20-30**  
**D-55411 Bingen**  
**Phone: +49 (0)6721 964-430**  
**Fax: +49 (0)6721 964-430**

.....  
**USA:**

**NSM Jukebox of America, Inc.**  
**1158 Tower Lane**  
**Bensenville, IL 60106**  
**Phone: +1 (630) 860-5100**  
**Fax: +1 (630) 860-5144**

**NSM**   
**JUKEBOX**  
A Company of NSM AG

Part-No.: xxx xxx Auf umweltfreundlichem, chlorfreiem Papier gedruckt.