

# Application Device Drivers

## Table of Contents

<b>CONVENTIONS .....</b>	<b>4</b>
<b>DEVICE DRIVERS .....</b>	<b>5</b>
Preliminaries .....	5
Phase .....	5
Phase For Outputs .....	5
Electrical Circuit Measurements .....	6
NORMAL Phase Reporting for Inputs .....	7
Reversed Phase Reporting for Inputs .....	7
<b>Digital Input Device Driver .....</b>	<b>8</b>
Data Items .....	9
Status and Event Codes .....	9
Commands And Responses .....	10
<b>Output Device Driver .....</b>	<b>10</b>
Data Item Definitions .....	11
Reporting States .....	11
Commands And Responses .....	11
<b>Reader Device Drivers .....</b>	<b>13</b>
<b>Floor Device Drivers .....</b>	<b>20</b>
<b>APPENDIX 1: DRAWINGS AND SCHEMATICS .....</b>	<b>21</b>
<b>APPENDIX 2: READER TECHNOLOGY PARAMETERS .....</b>	<b>28</b>
Reader Data Items .....	28
Reader Commands and Responses .....	28
Reader Data Acceptance, LED and Audio Enunciator Display Configurations .....	29
Reader Data Acceptance Configurations .....	32
<b>APPENDIX 3: READER OPERATIONS .....</b>	<b>34</b>
Reader Operations For Access Transactions .....	34
Badge Number Structure .....	37
Reader Operations For Command Transactions .....	37
Process Active State .....	38
Special Commands .....	38
The #1 Command – Close Area (Arm Zone) .....	39
The #2 Command – Open Area (Disarm Zone) .....	39
<b>APPENDIX 4: MESSAGE DECODING .....</b>	<b>40</b>
Bit Data Decoding for Badges .....	40
Character Data Decoding .....	41
<b>APPENDIX 5: READER TECHNOLOGY SPECIFICATIONS .....</b>	<b>43</b>
Display State Configuration .....	43
Key Pad Translations .....	43
Key Pad Data Decoding .....	44

<b>Reader Display Configuration Table .....</b>	<b>45</b>
Revision History .....	46

## List of Tables and Figures

Table 1: Conventions Used In This Document .....	4
Table 2: Status Reporting From A Digital Input Device Driver.....	9
Table 3: Digital Input Device Driver Commands & Responses.....	10
Table 4: Status Reporting From An Output Device Driver .....	11
Table 5: Relay Output Device Driver Commands & Responses .....	12
Table 6: Reader Technology Templates and Readers .....	13
Table 7: Card Decoding Rule Sets .....	15
Table 8: Supported Readers, Display Interfaces, and Decoding Interfaces.....	17
Figure 1: OSC Circuit Board Layout and Numbering Scheme .....	22
Figure 2: Base Board Numbering Scheme .....	23
Figure 3: Expansion Board Numbering .....	24
Figure 5: Wiegand Message Mask and Message Decoding .....	26
Figure 6: Encoded ID Decoding For Fixed Format Wiegand Cards.....	27
Table 9: Reader Commands and Responses.....	28
Table 10: Generic Reader HMI and Data Acceptance Specification.....	30
Table 11: Display Signal Codes (XXXX) Interpretation .....	31
Table 12: Data Acceptance Configurations .....	32
Table 13: Status Reporting From A Standard Reader .....	32
Table 14: Reader Display Specification Form.....	45

## Conventions

Within this document, the following text attribute conventions are used to represent certain types of information or items. The table identifies the information or item type, explains the display attributes, and provides examples.

**Table 1: Conventions Used In This Document**

Item	Display Convention
<b>PORT OR DRIVER STATUS</b>	The <b>STATUS</b> will be represented as uppercase, bold text. Examples include: <b>OPEN</b> <b>CLOSED</b> <b>READY</b>
<i>COMMAND, PROCESS OR METHOD OF A DEVICE DRIVER</i>	The <i>COMMAND, PROCESS, or METHOD</i> will be represented as upper case, italics text. Examples include: <i>REQUEST_PIN</i> <i>READY</i> <i>VALID_COMPLETE</i>
<i>any data item of a device driver</i>	All data items will be lower case italics with words concatenated with the underscore character (_). Examples include: <i>slow_pulse_interval</i> <i>start_msg_code</i> <i>begin_field_code</i> <i>format</i>
event codes	All event codes will be represented in lower case characters. Examples include: i4o i4s acs cmd

## Device Drivers

### Preliminaries

Device drivers are software modules installed in the Executive that provide the lowest application layer of software interface to the hardware physical ports on the Base Board and Expansion Board. Device drivers have the processing functions of interrogation for and reporting of status changes at the ports and the reception and transmission of messages or commands from or to the device. Message reception and transmission includes first level translation of messages in the case of readers. Further, in the case of some readers or other complex devices, a device driver may control more than one physical port.

Device drivers may play large or small computing roles in the application processes of the Open System Controller. Some may only report status changes while others may provide much more complicated responses to messages which represent transactions.

Device drivers generally have no memory for the specific data of a specific transaction. Some device drivers are designed to control a session of multiple transactions. A device driver that controls a session may retain temporary storage of information during a single session. This time period will be limited and information will not be retained beyond the end of a session.

All messages or transactions have a destination, a source, an event code (message type) and data values appropriate to the event code. When a device driver reports its status, it is reporting its current state, not the state from which it came.

### Phase

An important parameter which determines the operation of some device drivers is called *phase*. *Phase* determines how the device status of devices are reported or established. A device's *phase* may be either **NORMALLY\_OPEN (NO)** or **NORMALLY\_CLOSED (NC)**. *Phase* permits synchronization of the actual electrical state of a device with its operational reporting state.

### Phase For Outputs

The specification of *phase* establishes a context for the interpretation of commands to output's. Specifying *phase* lets the user define the actual state **ENERGIZED** or **DEENERGIZED** when the output is in its normal state. If an

output's phase is **NORMAL**, then the following interpretations will apply to the receipt of a change of status command.

- ✓ If the output has **NORMAL** phase, then a command to *OPEN* will **DEENERGIZE** the output.
- ✓ If the output has **NORMAL** phase, then a command to *CLOSE* will **ENERGIZE** the output.
- ✓ If the output is **DEENERGIZED** then it is **OPEN**.
- ✓ If the output is **ENERGIZED**, then it is **CLOSED**.

If the output's phase is **REVERSED**, then the following interpretations will apply to the receipt of a change of status command.

- ✓ If the output has **REVERSED** phase, then a command to *OPEN* will **ENERGIZE** the output.
- ✓ If the output has **REVERSED** phase, then a command to *CLOSE* will **DEENERGIZE** the output..
- ✓ If the output is **DEENERGIZED** then it is **CLOSED**.
- ✓ If the output is **ENERGIZED**, then it is **OPEN**.

### ***Electrical Circuit Measurements***

The specification of *phase* establishes the interpretation of an input's status in relation to the electrical circuit's measured condition (resistance). Specifying *phase* lets the user define the translation of **OPEN** or **CLOSED** into **ALARM** or **NORMAL**.

The following list defines the relationship between reported status and detected resistance levels for digital inputs.

- ✓ **OPEN\_CIRCUIT**: Infinite resistance detected.
- ✓ **SHORT\_CIRCUIT**: Zero resistance detected.
- ✓ **CLOSED**: Small resistance detected.
- ✓ **OPEN**: Large resistance value is detected.

### ***NORMAL Phase Reporting for Inputs***

If the input phase is said to be **NORMAL** (normally closed), then the following interpretation will apply to the evaluation of the electrical status of the input.

#### *Two State Inputs*

- ✓ If the input measures an **OPEN\_CIRCUIT** circuit, then the status is **ALARM**.
- ✓ If the input measures a **SHORT\_CIRCUIT** circuit, then the status is **NORMAL**.
- ✓ If the input measures a **CLOSED** circuit, then the status is **undefined since such a report is not possible from a two state device**.
- ✓ If the input measures a **OPEN** circuit, then the status is **undefined since such a report is not possible from a two state device**.

#### *Four State Inputs*

- ✓ If the input measures an **OPEN\_CIRCUIT** circuit, then the status is **OPEN\_CIRCUIT**.
- ✓ If the input measures a **SHORT\_CIRCUIT** circuit, then the status is **SHORT\_CIRCUIT**.
- ✓ If the input measures a **OPEN** circuit, then the status is **ALARM**.
- ✓ If the input measures a **CLOSED** circuit, then the status is **NORMAL**.

#### *Two State Reporting From Four State Devices*

- ✓ If the input measures an **OPEN\_CIRCUIT** circuit, then the status is **ALARM**.
- ✓ If the input measures a **SHORT\_CIRCUIT** circuit, then the status is **NORMAL**.
- ✓ If the input measures a **CLOSED** circuit, then the status is **HARDWARE\_ERROR**.
- ✓ If the input measures a **OPEN** circuit, then the status is **HARDWARE\_ERROR**.

### ***Reversed Phase Reporting for Inputs***

If the input phase is **REVERSED**, then the following interpretation will apply to the evaluation of the status of the input.

*Two State Inputs*

- ✓ If the input measures an **OPEN\_CIRCUIT** circuit, then the status is **NORMAL**.
- ✓ If the input measures a **SHORT\_CIRCUIT** circuit, then the status is **ALARM**.
- ✓ If the input measures a **CLOSED** circuit, then the status is **undefined since such a report is not possible from a two state device**.
- ✓ If the input measures a **OPEN** circuit, then the status is **undefined since such a report is not possible from a two state device**.

*Four State Inputs*

- ✓ If the input measures an **OPEN\_CIRCUIT** circuit, then the status is **OPEN\_CIRCUIT**.
- ✓ If the input measures a **SHORT\_CIRCUIT** circuit, then the status is **SHORT\_CIRCUIT**.
- ✓ If the input measures a **OPEN** circuit, then the status is **NORMAL**.
- ✓ If the input measures a **CLOSED** circuit, then the status is **ALARM**.

*Two State Reporting From Four State Devices*

- ✓ If the input measures an **OPEN\_CIRCUIT** circuit, then the status is **NORMAL**.
- ✓ If the input measures a **SHORT\_CIRCUIT** circuit, then the status is **ALARM**.
- ✓ If the input measures a **CLOSED** circuit, then the status is **HARDWARE\_ERROR**.
- ✓ If the input measures a **OPEN** circuit, then the status is **HARDWARE\_ERROR**.

## Digital Input Device Driver

The standard digital input device driver is assumed to be monitoring a single four state device. This device driver reports the status conditions defined in Table 2: Status Reporting From A Digital Input Device Driver below.

## Data Items

The parameters in the OSC Database Structure Reference Table define the digital input data items. The table name is INPDEF.

## Status and Event Codes

The proper calculation of reporting status and associated event code is defined in the table below. These are reports about transactions received by the device driver and reports in response to received questions or commands from another process. The table below shows all of the status reporting event codes supported.

**Table 2: Status Reporting From A Digital Input Device Driver**

Event	Send To Queue	Status	Phase (o,c)	Type (I,J)	Transition To Status
ihe	<i>send_queue</i>	HARDWARE ERROR	O	2STATE	OPEN
ihe	<i>send_queue</i>	HARDWARE ERROR	O	2STATE	CLOSED
ion	<i>send_queue</i>	ALARM	O	2STATE	SHORT_CIRCUIT
ioa	<i>send_queue</i>	NORMAL	O	2STATE	OPEN_CIRCUIT
ihe	<i>send_queue</i>	HARDWARE ERROR	C	2STATE	OPEN
ihe	<i>send_queue</i>	HARDWARE ERROR	C	2STATE	CLOSED
ica	<i>send_queue</i>	NORMAL	C	2STATE	SHORT_CIRCUIT
icn	<i>send_queue</i>	ALARM	C	2STATE	OPEN_CIRCUIT
jon	<i>send_queue</i>	NORMAL	O	4STATE	OPEN
joa	<i>send_queue</i>	ALARM	O	4STATE	CLOSED
jos	<i>send_queue</i>	SHORT_CIRCUIT	O	4STATE	SHORT_CIRCUIT
joo	<i>send_queue</i>	OPEN_CIRCUIT	O	4STATE	OPEN_CIRCUIT
jca	<i>send_queue</i>	ALARM	C	4STATE	OPEN
jcn	<i>send_queue</i>	NORMAL	C	4STATE	CLOSED
jcs	<i>send_queue</i>	SHORT_CIRCUIT	C	4STATE	SHORT_CIRCUIT
jco	<i>send_queue</i>	OPEN_CIRCUIT	C	4STATE	OPEN_CIRCUIT
ids	<i>log_queue</i>	DISABLED	(all)	(all)	DISABLED
ien	<i>log_queue</i>	ENABLED	(all)	(all)	ENABLED
ihe	<i>log_queue</i>	HARDWARE ERROR	(all)	(all)	HARDWARE ERROR
ise	<i>log_queue</i>	SOFTWARE ERROR	(all)	(all)	SOFTWARE ERROR

The actual reported event code will be the event code from the table above with the associated AEN specification appended to the event code. Thus if the value for *joo*, the AEN specification associated with event code *joo*, is "A", then the reported event code actually reported is A *jooa*. If the value of

joo was "E", the reported event would be joee and if the value was "N", the event would not be reported. In the case of ihe and ise, additional error data may be reported.

This device driver utilizes a Device Event, Type D, event log message format for all messages.

## Commands And Responses

The following commands will be accepted by the device driver and cause the indicated actions.

**Table 3: Digital Input Device Driver Commands & Responses**

Command	Response
<i>UPDATE</i>	The device driver will interrogate the data table that contains its configuration information and update its configuration to match that found in the database. It will synchronize its <b>ENABLED / DISABLED</b> status with the value determined by its schedule.  If the UPDATE command changes the status to <b>ENABLED</b> the device driver will report the event IEN and then report the current status of the device driver. If the UPDATE command changes the status to <b>DISABLED</b> the device driver will report the event ids.
<i>REPORT(process id)</i>	The current state of the device driver will be reported to the process requesting the report.
<i>ENABLE</i>	The device driver will place itself into the <b>ENABLED</b> status. It will retain this status until the next change of status from <b>ENABLED</b> to <b>DISABLED</b> or <b>DISABLED</b> to <b>ENABLED</b> dictated by changes from or to active periods of the schedule specified by the <i>schedule_id</i> in the device driver's configuration.
<i>DISABLE</i>	The device driver will place itself into the <b>DISABLED</b> status. It will retain this status until the next change of status from <b>ENABLED</b> to <b>DISABLED</b> or <b>DISABLED</b> to <b>ENABLED</b> dictated by changes from or to active periods of the schedule specified by the <i>schedule_id</i> in the device driver's configuration.
<i>(AUTO)</i>	Whenever the device driver detects a change of status, it will calculate the correct event code and report the new status to the process defined by the process id found in <i>send_queue</i> .

## Output Device Driver

The standard output device driver controls an OSC relay or digital output that is assumed to have both normally open and normally closed contacts. Thus, when the output is ENERGIZED or DE-ENERGIZED, it may be either OPEN or CLOSED. Establishing an explicit status requires a choice of phase. A digital output does not have both contact choices, however, for user convenience, it is assumed that it does since the controlled device may provide such a

capability or more conveniently be viewed in such a context. Phase has been defined for these devices above.

## Data Item Definitions

The parameters in the OSC Database Structure Reference Table define the digital output data items. The table name is OUTDEF.

## Reporting States

The device driver may be interrogated about the state of the device driver or the state of the relay. The proper calculation of reporting states is shown in the table below. The table below shows all of the reporting states that the device driver may supply.

**Table 4: Status Reporting From An Output Device Driver**

Event Code	Send To Queue	Status	Phase	Transition To Status
ood	<i>send_queue</i>	OPEN	n	DEENERGIZED
ooe	<i>send_queue</i>	CLOSED	n	ENERGIZED
ocd	<i>send_queue</i>	CLOSED	r	DEENERGIZED
oce	<i>send_queue</i>	OPEN	r	ENERGIZED
ods	<i>log_queue</i>	DISABLED	(all)	DISABLED
oen	<i>log_queue</i>	ENABLED	(all)	ENABLED
ohe	<i>log_queue</i>	HARDWARE ERROR	(all)	HARDWARE ERROR
ose	<i>log_queue</i>	SOFTWARE ERROR	(all)	SOFTWARE ERROR

The actual reported event code will be the event code from the table above with the associated AEN specification appended to the event code. Thus if the value for ood, the AEN specification associated with event code ood, is "A", then the reported event code actually reported is ooda. If the value of ood was "E", the reported event would be oode and if the value was "N", the event would not be reported. In the case of ohe and ose, additional error data may be reported.

The Device Event, Type D, message format is used for all of these messages.

## Commands And Responses

The following commands will be accepted by the device driver and cause the indicated actions.

**Table 5: Relay Output Device Driver Commands & Responses**

Command	Response
UPDATE	<p>The device driver will interrogate the data table that contains its configuration information and update its configuration to match that found in the database. It will synchronize its <b>ENABLED / DISABLED</b> status with the value determined by its schedule.</p> <p>If the UPDATE command changes the status to <b>ENABLED</b> the device driver will report the event IEN and then report the current status of the device driver. If the UPDATE command changes the status to <b>DISABLED</b> the device driver will report the event ods.</p>
ENABLE	<p>The device driver will place itself into the <b>ENABLED</b> status. It will retain this status until the next change of status from <b>ENABLED</b> to <b>DISABLED</b> or <b>DISABLED</b> to <b>ENABLED</b> dictated by changes from or to active periods of the schedule specified by the <i>schedule_id</i> in the device driver's configuration.</p>
DISABLE	<p>The device driver will place itself into the <b>DISABLED</b> status. It will retain this status until the next change of status from <b>ENABLED</b> to <b>DISABLED</b> or <b>DISABLED</b> to <b>ENABLED</b> dictated by changes from or to active periods of the schedule specified by the <i>schedule_id</i> in the device driver's configuration.</p>
OPEN( <i>duration, delay</i> )	<p>Calculate the correct command to instruct the output's device driver to open its contacts. This command will be ignored if the device is DISABLED. The interpretation will be based on the device's phase as downloaded. The commanded state will be maintained for a duration of <i>duration</i> after a delay of <i>delay</i>. If a duration of zero is specified, the duration will be assumed to be indefinite. A delay of zero means no delay.</p>
CLOSE( <i>duration, delay</i> )	<p>Calculate the correct command to instruct the output's device driver to close its contacts. This command will be ignored if the device is DISABLED. The interpretation will be based on the device's phase as downloaded. The commanded state will be maintained for a duration of <i>duration</i> after a delay of <i>delay</i>. If a duration of zero is specified, the duration will be assumed to be indefinite. A delay of zero means no delay.</p>
REPORT( <i>process id</i> )	<p>The DEVICE will report the DEVICE states found in Table 4: Status Reporting From An Output Device Driver to the process with <i>process_id</i> specified. A Type D record is transmitted.</p>
PULSE( <i>duration, delay</i> )	<p>This command directs the device driver to open the contacts of the output for the time interval <i>pulse</i>. This command has the same effect as the command sequence: {issue an OPEN command, wait for a time interval of <i>pulse</i> seconds, and then issue a CLOSE command}.</p>
SLOW_PULSE [ <i>duration, delay</i> ]	<p>This command directs the device driver to slowly and repetitively cycle the output from <b>DEENERGIZED</b> to <b>ENERGIZED</b> and back to <b>DEENERGIZED</b>, the for time interval <i>duration</i>. The time for each wait period is <i>slow_pulse</i>. The interpretation will be based on the device's phase as downloaded. The commanded state will be maintained for a duration of <i>duration</i> after a delay of <i>delay</i>. If a duration of zero is specified, the duration will be assumed to be indefinite. A delay of zero means no delay.</p>

Command	Response
<i>FAST_PULSE</i> [ <i>duration, delay</i> ]	This command directs the device driver to quickly and repetitively cycle the output from <b>DEENERGIZED</b> to energized and back to <b>DEENERGIZED</b> , output for the time interval <i>duration</i> . The time for each wait period is <i>fast_pulse</i> in seconds. The interpretation will be based on the device's phase as downloaded. The commanded state will be maintained for a duration of <i>duration</i> after a delay of <i>delay</i> . If a duration of zero is specified, the duration will be assumed to be indefinite. A delay of zero means no delay.

## Reader Device Drivers

The Reader Device is a template for all readers. Specific sets of data item values constitute implementations of this template for specific readers since all readers are not the same. The largest variation between readers will be in their display interface specification; the use of digital outputs to control LEDs and Audio (Beeper) outputs to implement the human machine interface and data acceptance states of the readers. Another variation between readers is the variation in the types of cards they can read. In fact, reader device drivers must be capable of decoding several types of card data encoding rules. This arises because of the ability of readers to read multiple formats.

The data structure of the reader device driver permits the specification of a display interface for each reader (a specification of how the reader is controlled and the details of its human machine interface called the reader technology and defined in the reader technology {RTKDEF} table) and the specification of up to five encoding formats (ECDDEF table records) for cards which might be presented to the reader. When multiple encoding formats are specified, the device driver sequentially processes the defined encoding formats and assumes that the first instance of an encoding format specification, which the card read transaction data satisfies, is assumed to be the desired format and no further evaluation of card formats will be attempted for that card. If no acceptable format is found, an icf (invalid card format) event is sent.

The following table provide an inventory of typical commercially available readers which will be supported in a default system load. Any of these may be adapted or modified by an informed user.

**Table 6: Reader Technology Templates and Readers**

Technology Template	Manufacturer	Model	Type
		HID Mini Prox 5365B	
		HID Mini Prox 5368B	
		HID Prox Pro 5352A	

		HID Prox Pro 5352A	
		HID Prox Pro 6030	
		HID Prox Point 6005	
		HID Prox Point 6008	
		HID ThinLine II 5365	
		HID ThinLineII 5368	
		HID Maxi Prox 5375	
		HID Tumstile 314	
		HID Epic 315	
		HID Insertion 312	
		HID PIN Pad 310	
		HID Classic Swipe 310	
		HID 5355	
		HID 5355 K	
		HID 5352A	
		HID 5352-485	
		HID 5365	
		HID 5353A485	
		HID 5352-232	
		HID 5638	
		HID 6005	
		HID 6008	
		HID 5398	
		HID 5395	
		HID 314	
		HID 315	
		HID 312	
		HID 310K	
		HID 310	
		Indala ARK-501SL	
		Indala ARK-501 PL	
		ARK-501 DL	
		Magtek MT-211232	
		SIASWR	
		Dorado Model 740	



DEMPI26I	MK	Dorado EMPI-II 26 bit	Encrypted codes are not decrypted.
DEMPI34	MK	Dorado EMPI-I 34 bit	Encrypted codes are not decrypted.
DEMPI34I	MK	Dorado EMPI-II 34 bit	Encrypted codes are not decrypted.
MS10F6I2	SS	Generic Magnetic Stripe	Encoded=10, Facility=6, Issue=2
SIA26	MK	SIA Standard 26 bit	

This last table indicates the type of reader, the display interface to be used, the valid decoding schemes, and the name of the applicable device driver for this reader type. Table 14: Reader Display Specification Form provides details on the display interface specification and Appendix 4: Message Decoding provides details on the decoding methodology. Type denotes the electrical / data encoding interface of the reader: S – Serial RS 232, W – Wiegand, C – Clock and Data, M – Serial RS 485 – 2 conductor.

**Table 8: Supported Readers, Display Interfaces, and Decoding Interfaces**

Reader Model	Version or Configuration	Type	Display Interface	Decode IF 1	Decode IF 2	Decode IF 3	Decode IF 4	Decode IF 5	Schematic
HID Mini Prox 5365B		WE							
HID Mini Prox 5368B		CD							
HID Prox Pro 5352A		23							
HID Prox Pro 5352A		48							
HID Prox Pro 6030		WE							
HID Prox Point 6005		WE							
HID Prox Point 6008		CD							
HID ThinLine II 5365		WE							
HID ThinLine II 5368		CD							
HID Maxi Prox 5375		WE							
HID Turnstile 314		WE							
HID Epic 315		WE							
HID Insertion 312		WE							
HID PIN Pad 310		WE							
HID Classic Swipe 310		WE							
HID 5355									
HID 5355 K									
HID 5352A									
HID 5352-485									
HID 5365									

Reader Model	Version or Configuration	Type	Display Interface	Decode IF 1	Decode IF 2	Decode IF 3	Decode IF 4	Decode IF 5	Schematic
HID 5353A485									
HID 5352-232									
HID 5638									
HID 6005									
HID 6008									
HID 5398									
HID 5395									
HID 314									
HID 315									
HID 312									
HID 310K									
HID 310									
Indala ARK-501SL		WE							
Indala ARK-501 PL		WE							
ARK-501 DL									
Magtek MT-211232		23							
SIASWR									
Dorado Model 740									
Dorado Model 712									

Reader Model	Version or Configuration	Type	Display Interface	Decode IF 1	Decode IF 2	Decode IF 3	Decode IF 4	Decode IF 5	Schematic
Dorado Model 780									
MM 6800-W									
MM6800-3									
SR-2400									
MT-211 232									
HID ProxPro Model 5355									
HID Prox Pro Model 5355K									
SIA 26 bit Standard	SIA 26 bit Standard								
AWID-MM-6800		WE							
AEWID-MM-6800		23							
AWID-SR-2400		WE							

## Floor Device Drivers

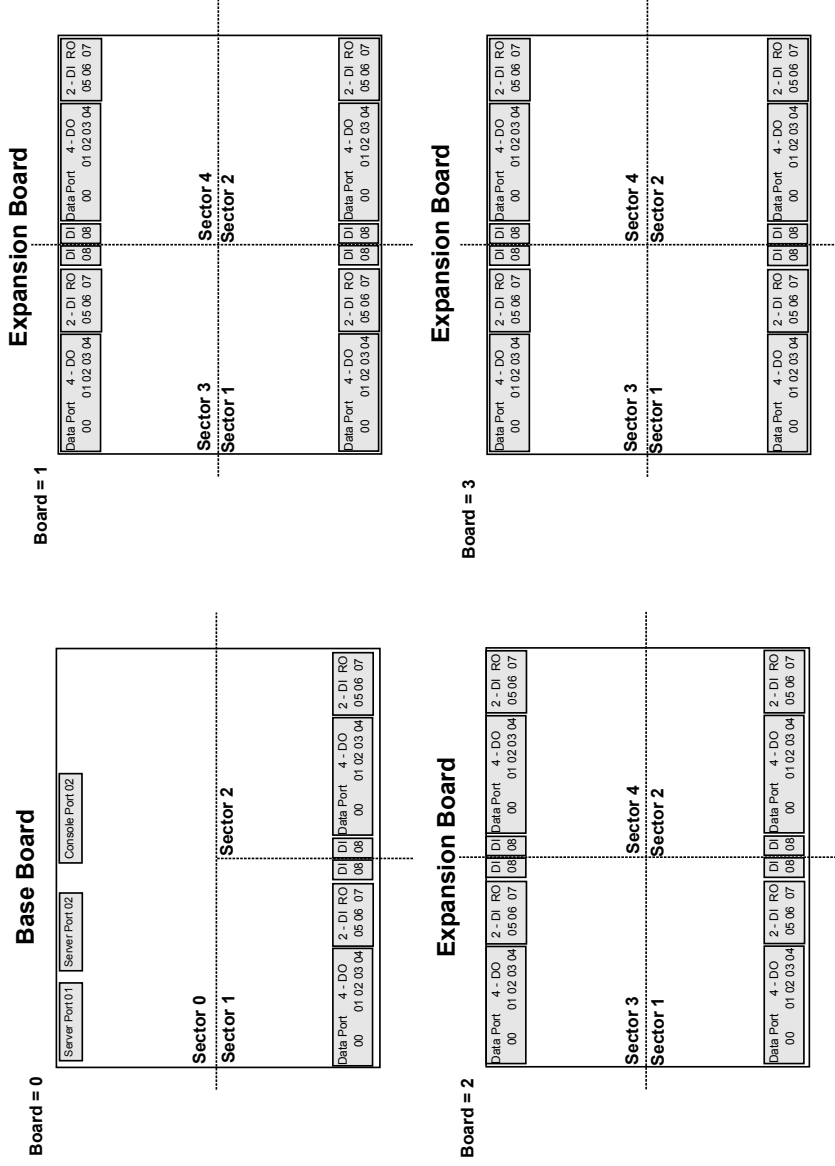
Version 1.2 of the Executive will implement a Floor device drive as a part of support for elevators. A floor device drive generally controls a digital input and a digital or relay output. The function in the process of the output is to enable an elevator button. The function of the digital input is to monitor what button has been pushed.

## Appendix 1: Drawings and Schematics

The following figures illustrate discussion in the test of this specification. The figures are:

1. **OSC Base and Expansion Board Layouts and Numbering Scheme:** This Figure describes the physical layout of the circuit boards of the OSC and the numbering scheme applied to that layout.
2. **Base Board Numbering Scheme:** This Figure describes the individual ports by type and the port number assignment made to them. It is for the base board only.
3. **Expansion Board Numbering:** This Figure describes the individual ports by type and the port number assignment made to them for expansion boards.
4. **Wiegand Parity Mask and Parity Checking:** This figure illustrated the relation ship between *parity\_mask*, *message\_mask*, and a sample message.
5. **Wiegand Message Mask and Message Decoding:** This figure illustrates how *message\_mask* can be used to decompose a Wiegand message into useful data.

Figure 1: OSC Circuit Board Layout and Numbering Scheme



ADDRESS IS: NNBSPP where NN = Node Number (10-99), B = Board Number (0-3), S = Sector (0-4), and PPP = Port Number (000 - 008). Please note that Sector 0 is on Board 0 Only.

Figure 2: Base Board Numbering Scheme

### Base Board Port Assignments

Port Name	Node	Board	Sector	Port No.	REL ADDR
Host Communications Port 1	NN	0	0	01	
Host Communications Port 2	NN	0	0	02	
Console Port	NN	0	0	03	
Tamper		0	0	05	
Data Port	NN	0	1	00	0
Digital Output	NN	0	1	01	1
Digital Output	NN	0	1	02	2
Digital Output	NN	0	1	03	3
Digital Output	NN	0	1	04	4
Digital Input 4S	NN	0	1	05	5
Digital Input 4S	NN	0	1	06	6
Relay Output	NN	0	1	07	7
Digital Input 4S	NN	0	1	08	8
Data Port	NN	0	2	00	10
Digital Output	NN	0	2	01	11
Digital Output	NN	0	2	02	12
Digital Output	NN	0	2	03	13
Digital Output	NN	0	2	04	14
Digital Input 4S	NN	0	2	05	15
Digital Input 4S	NN	0	2	06	16
Relay Output	NN	0	2	07	17
Digital Input 4S	NN	0	2	08	18

Note: REL ADDR (Relative Address) is, for the purpose of example, the relative address from the Sector 1 data port to the given port.

Figure 3: Expansion Board Numbering

Expansion Board Port Assignments

Port Name	Node	Board	Sector	Port No.	REL ADD R
Data Port	NN	1-4	1	00	0
Digital Output	NN	1-4	1	01	1
Digital Output	NN	1-4	1	02	2
Digital Output	NN	1-4	1	03	3
Digital Output	NN	1-4	1	04	4
Digital Input 4S	NN	1-4	1	05	5
Digital Input 4S	NN	1-4	1	06	6
Relay Output	NN	1-4	1	07	7
Digital Input4S	NN	1-4	1	08	8
Data Port	NN	1-4	2	00	10
Digital Output	NN	1-4	2	01	11
Digital Output	NN	1-4	2	02	12
Digital Output	NN	1-4	2	03	13
Digital Output	NN	1-4	2	04	14
Digital Input 4S	NN	1-4	2	05	15
Digital Input 4S	NN	1-4	2	06	16
Relay Output	NN	1-4	2	07	17
Digital Input 4S	NN	1-4	2	08	18

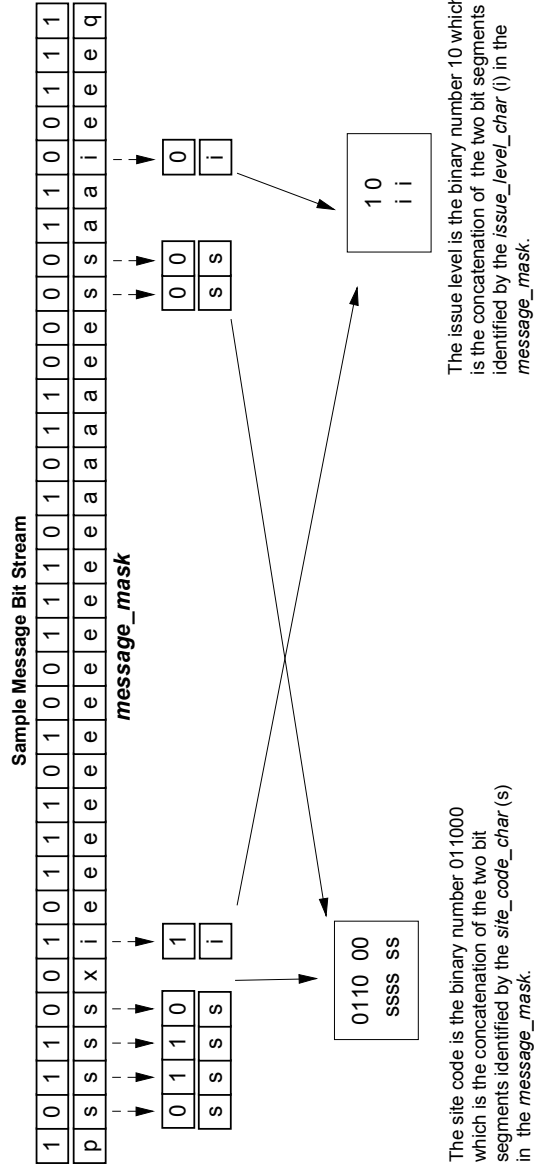
  

Port Name	Node	Board	Sector	Port No.	REL ADD R
Data Port	NN	1-4	3	00	20
Digital Output	NN	1-4	3	01	21
Digital Output	NN	1-4	3	02	22
Digital Output	NN	1-4	3	03	23
Digital Output	NN	1-4	3	04	24
Digital Input 4S	NN	1-4	3	05	25
Digital Input 4S	NN	1-4	3	06	26
Relay Output	NN	1-4	3	07	27
Digital Input4S	NN	1-4	3	08	28
Data Port	NN	1-4	4	00	30
Digital Output	NN	1-4	4	01	31
Digital Output	NN	1-4	4	02	32
Digital Output	NN	1-4	4	03	33
Digital Output	NN	1-4	4	04	34
Digital Input 4S	NN	1-4	4	05	35
Digital Input 4S	NN	1-4	4	06	36
Relay Output	NN	1-4	4	07	37
Digital Input 4S	NN	1-4	4	08	38

Note: REL ADDR (Relative Address) is, for the purpose of example, the relative address from the Sector 1 data port to the given port.



Figure 5: Wiegand Message Mask and Message Decoding



This figure illustrates the decomposition of the message into segments of bits which are concatenated (order preserving) into data items. The binary numbers thus generated may then be converted to decimal integers and stored in the respective fields as character representations of those integers. This example extracts the site code and the issue level. Here the issue level is 2 and the site code is 24. These decimal data values would be stored as characters in their respective data fields. They would be right justified and the fields left filled with the character zero (0).



## Appendix 2: Reader Technology Parameters

### Reader Data Items

The reader definition parameters in the OSC Database Structure Reference Table define the access control properties of a reader, the reader technology interface, and the card encoding formats to be evaluated by the reader device driver. The reader definition table name is RDRDEF. The display interface definition is in the table RTKDEF. Badge format decoding rules are defined in the table ECDDEF.

### Reader Commands and Responses

Reader device drivers implement the following commands. These commands may be effected within the device driver as a result of transaction processing or in response to receipt of the command.

**Table 9: Reader Commands and Responses**

Command	Response
UPDATE	The device driver will interrogate the table that contains its configuration information and update its configuration to match that found in the database.
REPORT(ADDRESS)	The device driver will report the state of the reader and associated digital outputs. If an address is specified, the WR will send the message to the specified address. Otherwise, it will send the report message to the send_queue address.
READY	This means place the Reader in the state defined as the Ready state in Table 10: Generic Reader where the properties of this state are determined by the reader's properties. Ready will open a closed port. If a transaction is pending, it will be cleared.
REQUEST_PIN	Set the reader's display to PIN Required and the data acceptance state to Accept PIN. If no PIN is received within <i>time_out_interval</i> , terminate the transaction as prescribed by the driver's operational specification.
REQUEST_BDG	Set the reader's display to Badge Required and the data acceptance state to Accept Badge. If no Badge is received within <i>time_out_interval</i> , terminate the transaction as prescribed by the driver's operational specification.
PIN_RETRY	Set the reader's display to Badge Required and the data acceptance display to Accept PIN. If no PIN is received within <i>time_out_interval</i> , terminate the transaction as prescribed by the driver's operational specification.
REQUEST_DATA	Set the reader's display to request command data from the keypad.

Command	Response
<i>DISABLE</i>	Disable the reader by setting disabled to True. Accept no data from the reader.
<i>ENABLE</i>	Enable the reader by setting disabled to False and placing the reader in the Ready State.
<i>VALID_COMP</i>	Place the reader in the Valid, Complete configuration. After time out, return the reader to the Ready display.
<i>INVALID_COMP</i>	Place the reader in the Invalid, Complete configuration. After time out, return the reader to the Ready display.
<i>READY_ALARM</i>	Place the reader in the Ready, Alarm configuration and place the reader in the Ready data acceptance configuration.
<i>READY_EXCEPT</i>	Place the reader in the Ready, Exception configuration and place the reader in the Ready data acceptance configuration.
<i>READY_TAMPER</i>	Place the reader in the Ready, Tamper configuration and place the reader in the Ready data acceptance configuration.
<i>PLEASE_WAIT</i>	Place the reader in the Please Wait configuration. This configuration times out in the time interval <i>processing_timeout</i> .
<i>RESET_PROCESS</i>	Instructs the driver to revert to the Ready state instead of the Process Active state at the conclusion of a session. This command sets <i>process_status</i> to False.
<i>PROCESS_ACTIVE</i>	Instructs the driver to switch to the Process Active state instead of the Ready state at the conclusion of a session. This command sets <i>process_status</i> to True.
<i>(AUTO)</i>	When the driver has received and decoded a complete message, it will send to the <i>send_queue</i> process the messages defined in Table 13: Status Reporting From A Standard Reader

## Reader Data Acceptance, LED and Audio Enunciator Display Configurations

The table provides the generic configuration of LED displays and audio annunciation supported by a generic OSC reader driver. The table defines the LED and Beeper actions used to signify the indicated states. These are the display states of the reader. The associated reader data acceptance state for each particular display state is also specified here. The data acceptance state is defined in Table 12: Data Acceptance Configurations.

Many readers will not support this specific configuration of LED functions. They may not have all of the LEDs or they may require different combinations of

outputs to control reader operation. The essential element of this table is the preservation of the relationship between display state and data acceptance state, not the actual colors used to represent a display state. Thus, when a reader does not support the generic color combination, the device driver will adapt to the reader's color scheme. Tables which specify the actual choices used for a specific readers will be found in a separate Appendix to this document: Reader Configuration Parameters.

The method of specifying the display or configuration state of a reader where such controls are accomplished by digital outputs follows the table below.

**Table 10: Generic Reader HMI and Data Acceptance Specification**

Display State	Data State	Description	Default XXXX
READY	<b>READY</b>	The reader is ready to receive either a badge or keypad entry depending on the reader's capabilities and default state.	<b>2222</b>
<i>DISABLE</i>	<b>DISABLED</b>	The reader has been disabled and will not respond to any instructions except Report, Update, and Ready All interim processing data is cleared	<b>2222</b>
VALID_COMP	<b>DISABLED</b>	The reader reports a valid transaction has occurred. All interim transaction data is cleared.	<b>2222</b>
PLEASE_WAIT	<b>DISABLED</b>	The reader reports that it is processing its last transaction and that the user should wait. Interim processing data is retained.	<b>2222</b>
PRO_ACTIVE	<b>(OPTION: DISABLED, READY)</b>	LED control may be assumed by another process by sending a specific display command. Typically this is used to signify that an associated door is unlocked. Interim processing data is retained.	<b>2222</b>
REQUEST_PIN	<b>ACCEPT PIN</b>	This configuration of lights indicates that the user should enter a PIN number. Interim processing data is retained.	<b>2222</b>
REQUEST_BDG	<b>ACCEPT BADGE</b>	This configuration of lights indicates that the user should use their badge. Interim processing data is retained.	<b>2222</b>
PIN_RETRY	<b>ACCEPT PIN</b>	This indicates that the PIN number entered was invalid and should be re-entered. Interim processing data is retained.	<b>2222</b>
REQUEST_DATA	<b>ACCEPT KEY DATA</b>	Set the reader's display to request data from the keypad for a user command.	<b>2222</b>

Display State	Data State	Description	Default XXXX
INVALID_COMP	<b>DISABLED</b>	This response represents a rejection of the transaction just attempted. All retained transaction data is cleared. The transaction must be restarted in its entirety after timeout of this state.	<b>2222</b>
READY_ALARM	<b>READY</b>	The reader is ready for business and is reporting an alarm condition at this site. All retained transaction data is cleared.	<b>2222</b>
READY_EXCEPT	<b>READY</b>	The reader is ready for business and is reporting an exception condition at this site. All retained transaction data is cleared.	<b>2222</b>
READY_TAMPER	<b>READY</b>	The reader is ready for business but is reporting a tamper condition. All retained transaction data is cleared.	<b>2222</b>

The following list provides values for the X parameters in the state specification above. A state specification requires that four values for X be specified. The following list of values are supported.

**Table 11: Display Signal Codes (XXXX) Interpretation**

Symbol	Meaning
0	OFF
1	ON STEADY
2	IGNORE
3	SLOW PULSE
4	FAST PULSE
A to E	1 TO 5 FAST PULSES
F to J	1 TO 5 SLOW PULSES

## Reader Data Acceptance Configurations

The following table defines the states into which the reader may be placed which characterize how it is to accept data.

**Table 12: Data Acceptance Configurations**

Command	Response
<b>ACCEPT BADGE</b>	Place the reader in the Accept Badge configuration. In some cases, this can require manipulation of the DOs that control LEDs and the Beeper. The meaning of this state is that only Badge data will be accepted.
<b>ACCEPT PIN</b>	Place the reader in the Accept PIN configuration. In some cases, this can require manipulation of the DOs that control LEDs and the Beeper. The meaning of this state is that only key pad data to be evaluated as a PIN will be accepted.
<b>ACCEPT KEY</b>	Place the reader in the Accept KEY configuration. In some cases, this can require manipulation of the DOs that control LEDs and the Beeper. The meaning of this state is that only key pad data will be accepted.
<b>DISABLED</b>	Place the reader in a disabled configuration. Accept no data from the reader. This does not mean that the device driver is disabled.
<b>READY</b>	The meaning of Ready is determined by the default configuration parameters. In some cases, this can require manipulation of the DOs that control LEDs and the Beeper. Usually it means accept either Badge or key pad data.

The following table inventories and describes all event codes used within the generic reader device driver. The details of these messages are found in the Open System Database Structure Reference. The message type (also event log message type) is found in the column titled type. Depending on the process which is creating or updating the event record, not all of the field values will be available. In that case, the field values which are not available should be blank filled when the record is formed.

**Table 13: Status Reporting From A Standard Reader**

Event	Status	Send To Queue	Type
acc	ACCESS EVENT	<i>access_queue</i>	A
ucm	USER COMMAND MESSAGE	<i>user_command_queue</i>	U
pto	PIN PAD TIME OUT	<i>log_queue</i>	A
cto	COMMAND TIME OUT	<i>log_queue</i>	A
cpe	CARD PARITY ERROR	<i>log_queue</i>	A
icf	INVALID CARD FORMAT	<i>log_queue</i>	A
rds	DISABLED	<i>log_queue</i>	D

---

<b>Event</b>	<b>Status</b>	<b>Send To Queue</b>	<b>Type</b>
ren	ENABLED	<i>log_queue</i>	D
rhe	HARDWARE ERROR	<i>log_queue</i>	D
rse	SOFTWARE ERROR	<i>log_queue</i>	D

## Appendix 3: Reader Operations

### Reader Operations For Access Transactions

Any occurrence of a hardware or software error detected by this device driver will be reported as an error as specified in Table 13: Status Reporting From A Standard Reader using the appropriate event codes, (rhe) and (rse) respectively.

The basic operation of the reader device driver is determined by the parameters defined in the RDRDEF (Reader Definition), RTKDEF (Reader Technology Definition), and the ECDDEF (Encoding Definition) tables on the controller. The objective of the reader's programming is to enable the collection of valid access request transactions in the form of a badge read, a badge read and PIN entry, or a PIN entry. Additionally, the reader device driver programming shall support the receipt and forwarding of a user-generated command, an area open (arm) request, and an area close (disarm) request. In all cases, the user is assumed to be informed, i.e., the user understands the key stroke and badge presentation protocol to be used to interact with the reader and knows all necessary user information. Such information includes, for example, a user's PIN number, the command syntax and the command data item values.

User interaction with the reader is organized into sessions. A session is a sequence of user – reader data exchanges (the reader blinks and beeps while the user types and swipes) in which a complete transaction is entered, an error occurs canceling the transaction, or the transaction entry times out, canceling the session. At the end of the session, any transaction data or counters accumulated during the session are cleared and reset to default values. All sessions start with the reader driver in the **READY** state.

An exception to this rule permits sessions to start when the reader state is **PRO\_ACTIVE**. If *process\_active* is true, command entry or access transactions may be initiated while the reader is in the **PRO\_ACTIVE** state. The choice of **PRO\_ACTIVE** or **READY** is determined by the value of *process\_status*. *Process\_active* describes a capability of the reader whereas *process\_status* describes the operational state of the reader. *Process\_status* is a local parameter. If false, the **READY** state is the session initiation state and, if true, the **PRO\_ACTIVE** state is the session initiation state. A common usage of this capability occurs when access transaction recording must continue at a reader when the associated portal is unlocked and shown that way by the reader's display of the **PRO\_ACTIVE** status.

The reader will not be responsive to the user and no session can be created if the reader driver is **DISABLED**. A reader is automatically **DISABLED** if the

current date and time do not fall within an active time of the *reader\_enable\_sched* and **ENABLED** if it does. A reader may be commanded to become **DISABLED** via the *DISABLE* command. The *DISABLE* command will cause the *cmd\_disable* flag to be set to True. The *ENABLE* command forces the reader to the **ENABLED** state also causing the *cmd\_disable* flag to be set to True. If *disable\_hold* is true, normal changes, in the state of the reader arising from changes in the *reader\_enable\_sched* will be ignored if *cmd\_disable* is True. The *disable\_hold* parameter permits the user the flexibility to determine whether *ENABLE* and *DISABLE* commands should expire with normal changes in reader schedule, or whether the commanded state should prevail. The *UPDATE* command resets the reader to its proper, scheduled status and resets *cmd\_disable* flag to false. Each change in the state (**ENABLED** or **DISABLED**) will be reported as specified in Table 13: Status Reporting From A Standard Reader using event codes (ren) and (rds) respectively.

Assuming that the driver is in the **READY** state (or **PRO\_ACTIVE**), then depending on the value of *concurrent\_pin*, the user may be required by the readers capabilities to start a transaction a specified way. Concurrent PIN says whether the reader can support key pad data capture concurrent with badge data capture without a change of state at the reader. If False, the state of the reader must be changed, usually by changes in DO configuration, for each type of data capture. If *concurrent\_pin* is false, then the value of *default\_mode* specifies the data acceptance state of **READY** and **PRO\_ACTIVE**. This specification should not be confused with *badge\_before\_pin*, which is used to require a specific order of data entry by the user. Assuming that concurrent PIN is false and that *default\_mode* is true, then the user must start a transaction with a badge read. PIN entries would be ignored. If *pin\_required* is false, then no PIN would ever be requested and PIN data would always be ignored. Note that this configuration would prevent user-initiated commands from being entered. *concurrent\_pin* and *default\_mode* fundamentally describe hardware configuration requirements and capabilities of a reader.

If *pin\_required* is true, a PIN will be required at those times that are within an active interval of the *pin\_active\_sched*. Similarly, if *badge\_required* is true, a badge entry will be required at those times that are within the active period of the *badge\_active\_sched*. The value of *pin\_required* and *badge\_required* are calculated and stored at the beginning of the session. Thus, the Ready state of the reader will vary over time as defined by the schedules *pin\_active\_sched* and *badge\_active\_sched*. An informed user will understand the operation of the reader and know the proper sequence for entering data. Note that PIN entry without Badge entry is presently undefined and cannot lead to successful access. This will be remedied by supporting a PIN only access method.

Readers that support concurrent badge and PIN functionality provide the most flexible operation for users. *badge\_before\_pin* defines whether there is a prescribed order. If this value is 0, there is no order. If the value is 1, badge must be presented before PIN and if the value is 2, PIN must be presented before Badge. This parameter is distinct from *default\_mode* which defines the readers hardware configuration but must be consistent with it. The usual default is badge first, then PIN, if required and requested by the reader.

When a PIN is required, the driver will count the number of PIN characters entered. If the number entered is less than the *pin\_length*, it will continue to prompt the user for another PIN character until it has accumulated the number of characters required for a valid PIN number. Subsequent entries will be ignored as long as the reader is in accept PIN mode. The driver indicates that it requires a PIN or another PIN character by placing the reader in the **REQUEST\_PIN** configuration. If the entry of PIN characters is so slow that the next character is not received within the *time\_out\_interval*, the transaction will be canceled and the driver will place the reader in the **INVALID\_COMP** configuration for a time period equal to the *time\_out\_interval*. The transaction message "PTO" will be reported. The driver will then clear all retained transaction data items and place the reader in the **READY** configuration.

Once a PIN of the correct length and badge, badge, or PIN only is entered, the driver places the reader in the **PLEASE\_WAIT** configuration. It formats and sends an access message to the process defined in the *send\_queue*. All of the data items included in the message are retained as temporary transaction data until there has been a processing timeout, an instruction to go to the **READY**, **INVALID\_COM**, **VALID\_COM**, or the **PIN\_RETRY** state, or the reader is **DISABLED**. After a time interval equal to *wait\_timeout*, the driver will change the configuration of the reader to the **INVALID\_COM** configuration. Typically, a valid entry will terminate the **PLEASE\_WAIT** state with a **READY**, **PRO\_ACTIVE**, **VALID\_COMPLETE**, **INVALID\_COMPLETE**, or **PIN\_RETRY** command before reaching this timeout.

The **PIN\_RETRY** command instructs the reader driver to give the user another chance to enter a valid PIN. Only *pin\_tries* retries to enter a valid PIN number are allowed. During a user session, receipt of a **PIN\_RETRY** command will cause the device driver to increment the *pin\_try\_count*. Upon receipt of the **PIN\_RETRY** command, the driver places the reader in the **PIN\_RETRY** Configuration. This prompts the user to enter a new PIN. The driver will count the number of PIN characters entered. If the number entered is less than the *pin\_length*, it will continue to prompt the user for another PIN character until it has accumulated the number required for a valid PIN number. Subsequent entries will be ignored as long as it is in PIN mode. The driver indicates that it requires a PIN or another PIN character by placing the reader in the **PIN\_RETRY** configuration. If the entry of a PIN character is so slow that more than the *time\_out\_interval* is exceeded, the transaction will be canceled and

the driver will place the reader in the **INVALID\_COM** configuration for a time period equal to the *time\_out\_interval*. The transaction message "PTO" will be sent. The driver will then clear all retained transaction data items and place the reader in the **READY** configuration.

When a complete PIN and badge or badge only is acquired by the driver, an access transaction message using the event code of (acc) is transmitted to the *access\_queue* process and the driver places the reader in the **PLEASE\_WAIT** state.

### Badge Number Structure

The most generic badge number involves three data items; *encoded\_id*, *facility\_code* (site code, customer code) and *bdg\_reissue\_no* (issue\_level). Of these, only the *encoded\_id* is required. When the device driver encounters a card specification which does not specify facility code or issue\_level, the device driver will supply default values of "000000" for facility code and "00" for issue\_level.

### Reader Operations For Command Transactions

The operation of the driver is similar in the case of commands. The reader must be in the **READY** or **PROCESS\_ACTIVE** state. The reader must be such that the key pad is active in the **READY** or **PROCESS\_ACTIVE** state if this feature is to be supported. The driver in the **READY** or **PROCESS\_ACTIVE** state recognizes the user's intent to enter a command because the first key it receives in this state is a *command\_start* character from the keypad. Users should choose this character so that it will not be confused with a data key entries and/or ARM/DISARM or OPEN/CLOSE commands.

A reader will not process commands if the time at the receipt of the *command\_start* character does not fall within the active portion of the *cmd\_enable\_sched* schedule. If no schedule is defined, commands will never be accepted.

Once the reader has entered command acceptance mode, it will first expect that a normal badge or badge and PIN access transaction will follow entry into the command mode. The process will collect this data and transmit it to the *user\_command\_queue* process defined within the reader's definition. This value is found in the CLRDEF record and is saved upon startup. It will remain in command mode and enter a **PLEASE\_WAIT** state and upon timeout, to **READY** or **PROCESS\_ACTIVE**. When it returns to **READY**, it will exit command mode.

During the **PLEASE\_WAIT** time out interval, the reader may be commanded to go to the **REQUEST\_DATA** state. There, it will continue to accept key pad

entries until *cmd\_max\_len* entries have been received or until it has received a *cmd\_field\_end* and *command\_end* character in sequence or until a time out of *time\_out\_interval* occurs. Again, users should choose these characters so that they will not be confused with other legitimate key pad data entries. Choosing *command\_end* to equal *cmd\_field\_end* works well. Key entries are ignored after this conditions has occurred and the reader will be placed in the **VALID\_COMP** state. Once that state has timed out, the reader will be placed into the **READY** state or, if **PROCESS\_ACTIVE** is true, the **PRO\_ACTIVE** state.

If, during character entry, the time interval between successive characters exceeds the *time\_out\_interval*, then the transaction will be canceled, the reader will be placed in the **INVALID\_COMP** state and, upon that state's time out, be placed in the **READY** or **PROCESS\_ACTIVE** state. The transaction message "cto" will be sent

Once the key pad has captured the correct number of characters or the character entry is terminated with the correct end of command sequence, the reader will be placed in the **PLEASE\_WAIT** state until it times out. It is then returned to the **READY** or **PRO\_ACTIVE** state. The process will transmit user command message as prescribed in Table 13: Status Reporting From A Standard Reader.

### Process Active State

The **PRO\_ACTIVE** state is activated by the receipt of *PRO\_ACTIVE* command, the *process\_status* flag will be set to True. If the reader is in the **READY** state, it will be placed in the **PRO\_ACTIVE** state. Upon receipt of the *Reset Process* command, the driver will set the *process\_status* flag to False. If the reader is in the **PRO\_ACTIVE** state it will be placed in the **READY** state.

### Special Commands

The following keypad commands are special and are reserved. Special commands begin with a "#" key entry. If the *command\_start* character is "#", these commands will never be available. There must be no current session for these commands to be recognized.

The operation of the driver for special commands is similar to the case of user commands. The reader must be in the **READY** or **PRO\_ACTIVE** state. The reader must be such that the key pad is active in the **READY** or **PRO\_ACTIVE** state if this feature is to be supported. The driver in the **READY** or **PRO\_ACTIVE** state recognizes the user's intent to enter a command because the first key it receives in this state is a "#" character from the keypad.

**The #1 Command – Close Area (Arm Zone)**

This command is used to close an area. When the driver is in any ready state ( **READY** or **PRO\_ACTIVE**) and a session is initiated with a key pad “#” followed by a “1”, it will initiate an access session after following the procedure defined in this paragraph. If the *arm\_disarm\_auth* flag is true, *arm\_disarm* will be set to 1 during the session and reset to 0 at the end of the session.

**The #2 Command – Open Area (Disarm Zone)**

This command is used to open an area. When the driver is in any ready state ( **READY** or **PRO\_ACTIVE**) and a session is initiated with a key pad “#” followed by a “2”, it will initiate an access session after following the procedure defined in this paragraph. If the *arm\_disarm\_auth* flag is true, *arm\_disarm* will be set to 2 during the session and reset to 0 at the end of the session.

## Appendix 4: Message Decoding

The following paragraphs define the various schemes for decoding badge encoded data.

The receipt of data through the RS232 and RS485 ports is standard in the sense that character formation is controlled in a fashion defined by the typical parameters for such data flows. Characters are received. The Wiegand and Clock & Data interfaces receive groups of bits that are not generally character oriented. This presents an additional level of complexity in the decoding of data stored in such cards. This case is discussed in the section Bit Data Decoding.

### Bit Data Decoding for Badges

Many schemes for encoding data are possible and used in this situation. Once the message bits are captured, messages must be validated and broken into components representing actual data items.

Wiegand and Clock & Data messages are simple because they tend to be short and complex because each manufacturer of readers and cards may utilize different data encoding schemes designated to be proprietary. A single reader may support multiple formats and within any encoding format, different rules for decoding the data portion of the message may be used. Thus, a considerable effort toward message analysis may be required.

The conventional Wiegand bit encoding formats are either 26 or 34 bit structures that begin and end with a parity bit. After those bits are removed, 24 and 32 bits are reserved for data. The values of the parity bits are determined so that an even and odd parity result will be calculated for different portions of the card's bit stream.

The first half of the message will be one parity, usually even, while the second half will be the opposite, typically odd. This allows the direction of swipe of the card to be determined. The data item `even_first` defines which is expected. If it is 0, there is no prescription for first, if it is 1, even first is prescribed, and if it is 2, odd first is prescribed. The standard (and default), such as it is, is even first.

If it is specified that the even segment is read before the odd segment in a normal read of the badge, then calculation of the parity of the first and last segments of the card's bits defines whether the card was read even before odd. When the device driver detects a reverse direction read, it should automatically reverse the order of the bit stream prior to decoding the data. This of course presumes that there were no errors in the badge's read process and no card parity errors, other than reversal, were detected. Other parity

errors are reported as a "cpe" error. Since odd length encoded data streams or unequal length parity mask patterns are generally not unequivocal, such patterns in Wiegand and Clock & Data formats are rarely seen. This also permits the software to accept bi-directional reads as most of the readers permit. Wiegand data decoding is dependent on bit stream order so detecting the direction of read and the correct order for interpreting the bits is essential.

Figure 4: Wiegand Parity Mask and Parity Checking graphically presents the decoding of the bit stream for the purpose of parity verification and card read order.

The basic Wiegand reader encodes data as integer values. Typically, the bit pattern may be decomposed into a series of multi-bit subsets that represent numbers. One field is usually called a facility code (site code, customer code) and one field is called the encoded number. In general use, each badge for a particular customer has the same facility code and a different encoded id. Card manufacturers control facility codes as a means of preventing cards at one site from being used at another. 26 bit cards (24 data bits) typically have 8 bits for facility codes (255 codes) and 16 bits (65535 values) for encoded ids while 34 bit cards (32 data bits) have 16 bits for facility codes (65535 codes) and 16 bits for encoded ids. Decompositions into other sub-fields are also possible. This decomposition is described by the *message\_mask*. The use of each bit position in a message bit stream is defined by the character value in the corresponding position on the *message\_mask*.

Card bit patterns are compared with masks that define the decomposition of the bit stream into sub-fields so that the individual sub-fields may be interpreted as binary numerical values. The device driver performs this decomposition and translates these values into data values for reporting to its supervising control process.

The data items *site\_code\_char*, *aux\_field\_char*, *encoded\_id\_char*, *issue\_level\_char*, and *ignore\_char* are used to define this encoding. These characters define which bits are to be collected into a group (using an order preserving process) and treated as a binary number. These binary numbers are then turned into characters and inserted into the appropriate data field where the data strings are justified and filled according to the field's specification.

### **Character Data Decoding**

Character data decoding is easier. A stream of data is captured from the card. The stream is then parsed in accordance with the rules prescribed by the cards encoding data items specification.

Badge encoded data may or may not have a starting or ending marker (sentinel) and may or may not have beginning or ending field marks (sentinels). *start\_msg\_code*, *end\_msg\_code*, *begin\_field\_code*, and *end\_field\_code* are used for this purpose. When fields are marked by sentinels, the sequence to expect a field must be specified. *seq\_cksum*, *seq\_enclid*, *seq\_site*, *seq\_il*, and *seq\_aux* are used for this purpose. The ordinality of their appearance, with 0 denoting absence, is their value. The "seq..." fields also indicate the expected presence of these fields in an encoding format.

Some encoding schemes are defined by position in the encoded data stream. The fields have start positions and ending positions. Some encoding schemes support checksums of different types (*ck\_sum\_type*).

## Appendix 5: Reader Technology Specifications

Several items in the data definition of the reader's technology specify the means by which the reader's display characteristics will be implemented. These fields and their use are discussed below.

### ***Display State Configuration***

Readers state is usually determined by the states of digital outputs connected to the reader. The OSC supports up to four such digital outputs. In the RTKDEF record, the data items `do_port_1`, `do_port_2`, `do_port_3`, and `do_port_4` are used to specify the availability of these ports for a particular technology. The technology process then knows which ports are to be avoided. This is important because the associated portal process may need to use the ports.

The most critical feature of the interface is maintenance of the display interface and data acceptance state. Thus, one must specify specific states for accept badge and accept PIN. Using a 0 to indicate de-energized, a 1 to indicate energized, and a 2 to indicate "don't care", the configuration of the four DOs to attain this state may be specified. This is done with the data items `accept_badge` and `accept_pin`. If `accept_badge = "0000"`, then all DOs OFF (CLOSED) are required to accept a badge. If `accept_badge = "1010"`, then DOs 1 and 3 must be ON (OPEN) while 2 and 4 are OFF (CLOSED). If `accept_badge = "2222"`, then DO states are not relevant to the state.

Once these assignment are complete the designer may establish patterns of visual display which are non-conflicting or supportive of with the data acceptance state.

### ***Key Pad Translations***

An additional decoding problem is encountered when a PIN pad is associated with such a reader. In some cases both data input means, badge and pad, are active at the same time. In other cases, the controlling software must establish the active data input means through the use of DOs or commands. The reader driver must be aware of the current context of the reader's data acceptance state to properly interpret the bits received from the reader.

Assuming that this context is clear (see Appendix 3: Reader Operations), then typically one key stroke at a time will be received. Different readers will produce different patters for key strokes. Received bits will be left justified in

an eight bit field and evaluated by comparing the field with the values stored for the keys supported by the reader. Based on this match, the proper key value may be determined. The translated value will be recorded in the access message created by the transaction.

### Key Pad Data Decoding

Parameter	Len	Value
<i>key_bit_len</i>	1	Length of key stroke data in bits (1 key stroke)
<i>key0</i>	1	Default Value is 11110000. (0xF0)
<i>key1</i>	1	Default Value is 11100001. (0xE1)
<i>key2</i>	1	Default Value is 11010010. (0xD2)
<i>key3</i>	1	Default Value is 11000011. (0xC3)
<i>key4</i>	1	Default Value is 10110100. (0xB4)
<i>key5</i>	1	Default Value is 10100101. (0xA5)
<i>key6</i>	1	Default Value is 10010110. (0x96)
<i>key7</i>	1	Default Value is 10000111. (0x87)
<i>key8</i>	1	Default Value is 01111000. (0x78)
<i>key9</i>	1	Default Value is 01101001. (0x69)
<i>key_star</i>	1	Default Value is 01011010. (0x5A)
<i>key_pound</i>	1	Default Value is 01001011. (0x4B)

## Reader Display Configuration Table

The table below illustrated how a reader's requirements for DO control may be documented and parameterized. The DO requirements for data acceptance state is specified. In fact there are only two that may be specified since disabled is implemented by the device driver... it simply ignores the reader..., accept badge and accept pin. A value of "2222" means no DO control is required as shown below for disabled. Then, the designer may indicate the DO configurations for the supported colors. Once the correct values for colors and data acceptance states are entered, the designer may establish non-conflicting display patterns for the display state. In some cases, less detail than envisioned in the generic standard may be required.

**Table 14: Reader Display Specification Form**

Display State	Data State	Generic State Configuration	DO Setting For State Configuration
READY	READY	2222	
DISABLE	DISABLED	2222	2222
VALID_COMP	DISABLED	2222	2222
PLEASE_WAIT	DISABLED	2222	2222
PRO_ACTIVE	DISABLED	2222	
PRO_ACTIVE	READY	2222	
REQUEST_PIN	ACCEPT PIN	2222	
REQUEST_BDG	ACCEPT BADGE	2222	
REQUEST_DATA	ACCEPT KEY	2222	
PIN_RETRY	ACCEPT PIN	2222	
INVALID_COMP	DISABLED	2222	2222
READY_ALARM	READY	2222	
READY_EXCEPT	READY	2222	
READY, TAMPER	READY	2222	

**Revision History**

<b>Date</b>	<b>Revision #</b>	<b>Specific Information About Revision</b>	<b>Pages/ Sections Affected By Revision</b>	<b>Authorizing Document or Action</b>
8/29/01	1.1	Changed document name to DPC630_OSC_Device_Drivers.doc	all	SK