

合肥金涵电子

-----<http://www.jinhandz.cn>

-----<http://www.jinhandz.com>

JHZK 库使用说明 V5.2



版本信息

版本	日期	修改说明
V5.2	2018/3/6	1. 新增<二维码节目-JHCreateQRCodeProg>用于指定分区创建二维码节目; 2. 新增<二维码节目指令-JHCreateQRCodeProgIst>用于批处理创建二维码节目;
V5.2	2018/1/15	1. 新增<恢复系统出厂设置指令-JHRestoreSysFactory>用于恢复整个系统出厂时参数和状态;
V5.2	2017/8/23	1. 新增<设置 I0 输出状态-JHSetIOState>用于设置 I0 口输出状态; 2. 新增<设置 I0 输出状态指令 - JHSetIOStateIst>用于批处理文件中添加设置 I0 输出状态指令;
V5.2	2017/8/17	1. 新增<创建高级数字时钟 -JHCreateComplexDigClockProg>用于添加高级数字时钟节目; 2. 新增<创建高级数字时钟指令 - JHCreateComplexDigClockProgIst>用于批处理文件中添加高级数字时钟指令; 3. 新增<设置语音参数- JHSpeechSynthesisSetPara > 用于设置语音参数; 4. 新增<设置语音参数指令 - JHSpeechSynthesisSetParaIst>, 用于批处理文件中添加设置语音参数指令;
V5.2	2016/3/10	1. 新增<开始语音合成指令- JHSpeechSynthesisStartIst> 用于批处理文件中添加开始语音合成指令; 2. 新增<停止语音合成指令- JHSpeechSynthesisStopIst> 用于批处理文件中添加终止语音播放指令; 3. 新增<暂停语音合成指令- JHSpeechSynthesisPauseIst> 用于批处理文件中添加暂停语音播放指令; 4. 新增<恢复语音合成指令-JHSpeechSynthesisRestartIst >, 用于批处理文件中添加恢复语音播放指令; 5. 新增<休眠语音模块指令- JHSpeechSynthesisSleepIst> 用于批处理文件中添加休眠语音模块指令; 6. 新增<唤醒语音模块指令- JHSpeechSynthesisWakeupIst >, 用于批处理文件中添加唤醒语音模块指令; 7. 新增<立即显示图片指令- JHDrawBitmapIst>, 用于批处理文件中添加立即显示图片指令;

V5.1	2016/3/5	<ol style="list-style-type: none"> 1. 新增<创建图片节目指令- JHCreateBitmapProgIst>, 用于批处理文件中添加图片节目质量; 2. 新增<创建倒计时指令- JHCreateCountDownProgIst>, 用于批处理文件中添加倒计时节目指令; 3. 新增<创建正计时指令- JHCreateCountUpProgIst>, 用于批处理文件中添加正计时节目指令; 4. 新增<开始语音合成- JHSpeechSynthesisStart>, 用于播放合成的语音文本; 5. 新增<停止语音合成- JHSpeechSynthesisStop>, 用于终止语音播放; 6. 新增<暂停语音合成- JHSpeechSynthesisPause>, 用于暂停语音播放; 7. 新增<恢复语音合成- JHSpeechSynthesisRestart>, 用于恢复暂停的语音播放; 8. 新增<查询语音模块- JHSpeechSynthesisInquire>, 查询语音模块当前状态; 9. 新增<休眠语音模块- JHSpeechSynthesisSleep>, 用于休眠语音模块; 10. 新增<唤醒语音模块- JHSpeechSynthesisWakeup>, 用于唤醒语音模块;
V5.0	2016/2/15	<ol style="list-style-type: none"> 1. 修改<显示图片 - JHDrawBitmap>, 修改图片源信息 (BMPSRC) 格式以及位图格式 format; 2. 修改<创建图片节目 - JHCreateBitmapProg>, 修改图片源信息 (BMPSRC) 格式以及图片格式 format;
V4.7	2016/1/13	<ol style="list-style-type: none"> 1. 新增<获取图片数目—JHGetBitmapCnt>, 用于获取当前图片的数目; 2. 新增<获取图片 ID—JHGetBitmapId>, 用于获得图片的 ID 号; 3. 新增<获取图片信息—JHGetBitmapInfo>, 用于获取图片信息; 4. 新增<查询图片是否存在—JHInquireBitmap>, 用于查询图片是否存在; 5. 新增<下载图片数据第一步发送图片数据总大小—JHPreCreateBitmap>, 用于发送图片数据大小; 6. 新增<下载图片数据第二步下载图片数据—JHWriteBitmap>, 用于下载图片数据; 7. 新增<下载图片数据第三步结束下载, 释放资源—JHFinCreateBitmap >, 用于释放图片资源; 8. 新增<删除图片文件—JHDeleteBitmap >, 用于删除图片文件;
V4.5	2015/11/19	<ol style="list-style-type: none"> 1. 修改版本号 V4.5 和指令文档一致; 2. 修改<获取默认节目—JHFingerProg>, 修改节目类型 (ptype); 修改正/倒计时节目对应 CONTENT 表格;

V1.9	2015/11/18	<ol style="list-style-type: none"> 1. 新增<立即显示图片节目—JHDrawBitmap>,用于在屏幕任意位置显示静态图片; 2. 新增<创建图片节目—JHCreateBitmapProg>,用于显示有特效的静态图片; 3. 新增<倒计时节目—JHCreateCountdownProg>,用于显示倒计时; 4. 新增<正计时节目—JHCreateCountupProg>,用于显示正计时;
V1.8	2015/11/05	<ol style="list-style-type: none"> 1. 修改通用节目属性<rev 和 style>,支持多节目和队列加载或立即加载; 2. 修改<JHCreateGridProg —flag>,支持表格节目停留时间; 3. 修改<JHDeleteProg — option>,支持删除分区指定节目编号节目; 4. 删除 JHGridSetEraseFlag 和 JHGridGetEraseFlag;

一. 步骤:

1. 调用 JHCreateInst/JHCreateRelativeInst/JHCreateAbsoluteInst 函数创建实例句柄;
2. 调用 JHMountTransportLayer 函数配置传输层;
3. 调用各种功能 APIs 函数, 如 JHGetGID, JHSetGID 等;
4. 调用 JHDeleteInst 函数删除实例句柄;

提示: 开发时一定要根据步骤调用, 否则可能会存在未知的风险, 内存泄露等。

二. 特别说明:

1. JHCreateInst(), JHCreateRelativeInst(), JHCreateAbsoluteInst() 的区别是:
JHCreateInst(): 在不知道卡的控制卡屏号的时候创建实例句柄;
JHCreateRelativeInst(): 在知道控制卡屏号的时候创建实例句柄;
JHCreateAbsoluteInst(): 在不知道控制卡屏号, 但知道控制卡产品 ID 的时候创建实例句柄;

提示: 上述三种创建方式中最好运用 JHCreateInst() 来创建。

2. JHMountTransportLayer

参数ptl: 串口通讯用&transportlayer_serialport (JHComm.h中),
广域网通讯用&transportlayer_wan (JHComm.h中);
参数argstruct: 串口通讯用LPINFO_SERIALPORT (&INFO_SERIALPORT) (JHComm.h),
广域网通讯用LPINFO_WAN (&INFO_WAN) (JHComm.h中);

提示: 参数ptl和参数argstruct也可以有用户自己编写代码

3. INFO_SERIALPORT

```
typedef struct tagInfo_SerialPort{  
    unsigned long size;  
    char serialport[64];  
    unsigned long baudrate;  
    unsigned char databits;  
    unsigned char parity;  
    unsigned char stopbits;  
    unsigned char flowctrl;  
    unsigned long timeout;  
} INFO_SERIALPORT, *LPINFO_SERIALPORT;
```

参数:

size: 大小。取值sizeof(INFO_SERIALPORT);

serialport: 串口。取值如 'Com1', 'Com2'。

baudrate: 波特率。取值9600, 14400, 19200, 38400, 56000, 57600, 115200。

databits: 数据位。取值8,9。
parity: 校验位。取值0,1,2。
stopbits: 停止位。取值0,1,2。
reserved: 保留位。取值0。
timeout: 超时。取值5000。

4. INFO_WAN

```
typedef struct tagInfo_Wan{  
    unsigned long size;  
    unsigned char usedomain;  
    unsigned char addr[67];  
    unsigned short port;  
    unsigned short reserved;  
    unsigned long timeout;  
} INFO_WAN, *LPINFO_WAN;
```

参数:

size: 大小。取值sizeof(INFO_WAN);
usedomain: 域名。取值0,1。
addr: 控制卡IP。取值如“192.168.16.254”。
port: 端口。取值如30000。
reserved: 保留位。取值0。
timeout: 超时。取值5000。

目录

1. JHCreateInst（创建实例句柄）	12
2. JHCreateRelativeInst（创建相对实例句柄）	12
3. JHCreateAbsoluteInst（创建绝对实例句柄）	13
4. JHChangeInstServerId（修改源地址）	13
5. JHChangeInstRelativeId(修改相对地址(屏号))	14
6. JHChangeInstAbsoluteId(修改绝对地址（产品 ID）).....	14
7. JHDeleteInst（删除实例句柄）	15
8. JHMountTransportLayer（配置传输层）	15
9. JHGetGID（查询相对地址(屏号)）	16
10. JHSetGID（配置相对地址(屏号)）	16
11. JHGetUID（查询绝对地址（产品 ID））	17
12. JHGetGIDByUID(通过绝对地址（产品 ID）查询相对地址(屏号))	17
13. JHSetGIDByUID（通过绝对地址（产品 ID）配置相对地址(屏号)）	18
14. JHGetSerialAttr(查询串口参数)	18
15. JHSetSerialAttr（配置串口参数）	19
16. JHGetPort（查询 IP 端口号）.....	19
17. JHSetPort（配置 IP 端口号）	20
18. JHGetIPv4（查询 IPv4）.....	20
19. JHSetIPv4（配置 IPv4）	21
20. JHGetMac（查询 MAC）.....	22
21. JHGetLedSize(查询屏宽屏高).....	22
22. JHSetLedSize（配置屏宽屏高）	22
23. JHGetDefLedMod（查询默认单元板参数）	23
24. JHSetDefLedMod（配置默认单元板参数）	24
25. JHGetCurLedMod（查询当前单元板参数）	26
26. JHSetCurLedMod（配置当前单元板参数）	27
27. JHGetDefRefreq（查询扫描频率）	29
28. JHSetDefRefreq(配置扫描频率).....	29
29. JHGetDefLum（查询默认亮度）	30
30. JHSetDefLum（配置默认亮度）	30
31. JHGetCurLum（查询当前亮度）	30
32. JHSetCurLum（配置当前亮度）	31
33. JHGetDefFont(查询默认字库文件).....	31
34. JHSetDefFont(配置默认字库文件)	32
35. JHGetCurFont(查询当前字库文件).....	32
36. JHSetCurFont（配置当前字库文件）	32
37. JHGetDefPenColor（查询默认画笔颜色）	33
38. JHSetDefPenColor（配置默认画笔颜色）	33
39. JHGetCurPenColor（查询当前画笔颜色）	34
40. JHSetCurPenColor（配置当前画笔颜色）	34
41. JHGetDefBrushColor（查询默认画刷颜色）	35
42. JHSetDefBrushColor（配置默认画刷颜色）	35

43. JHGetCurBrushColor（查询当前画刷颜色）	36
44. JHSetCurBrushColor（配置当前画刷颜色）	36
45. JHGetLocale(查询区域语言).....	37
46. JHGetSystemTime（查询系统时间）	37
47. JHSetSystemTime（配置系统时间）	38
48. JHGetSystemVersion（查询系统版本号）	39
49. JHGetRestCapacity（查询 flash 剩余空间）	39
50. JHGetTotalCapacity（查询 flash 空间总大小）	39
51. JHGetSilentMode（查询静默模式）	40
52. JHSetSilentMode（配置静默模式）	40
53. JHPower（开关机）	41
54. JHRestart（重启）	41
55. JHPreUpgrade（升级第一步发送升级数据总大小）	41
56. JHWriteUpgrade（升级第二步发送升级数据）	42
57. JHFinUpgrade（升级第三步结束升级，释放资源）	43
58. JHRestoreFactory（恢复参数出厂设置）	43
59. JHRestoreSysFactory（恢复系统出厂设置）	44
60. JHGetFontCnt（查询字体数目）	44
61. JHGetFontId(查询字体 ID).....	44
62. JHGetFontInfo(查询字体详细信息).....	45
63. JHGetFontInfoEx(高级查询字体详细信息).....	46
64. JHInquireFont（查询字体是否存在）	46
65. JHPreCreateFont（下载字库数据第一步发送字库数据总大小）	47
66. JHWriteFont（下载字库数据第二步发送字库数据）	47
67. JHFinCreateFont（下载字库数据第三步结束下载，释放资源）	48
68. JHDeleteFont（删除字库文件）	49
69. JHErase(清屏).....	49
70. JHDrawMultPixel（点亮点组）	49
71. JHDrawMultPixelEx（高级点亮点组）	50
72. JHDrawLine（画线）	50
73. JHDrawLineEx（高级画线）	51
74. JHFrameRect（画矩形）	52
75. JHFrameRectEx（高级画矩形）	52
76. JHFillRect（填充矩形）	53
77. JHFillRectEx（高级填充矩形）	53
78. JHDrawText（立即显示文本）	54
79. JHDrawBitmap（立即显示图片）	55
80. JHScrollLeft（左移一格）	57
81. JHScrollRight（右移一格）	57
82. JHScrollUp（上移一格）	58
83. JHScrollDown（下移一格）	58
84. JHGetWndCnt（查询分区数目）	59
85. JHGetWndId（查询分区编号）	59
86. JHFingerWnd（查询分区属性）	60

87. JHCreateWnd（创建分区）	60
88. JHChangeWnd（修改分区属性）	61
89. JHDeleteWnd（删除分区）	62
90. JHGetProgress（查询分区播放状态）	62
91. JHCreateTextProg（创建文本节目）	63
92. JHCreateBitmapProg（创建图片节目）	66
93. JHCreateQRCodeProg（创建二维码节目）	68
94. JHCreateSimpleDigClockProg（创建简易时钟）	71
95. JHCreateDigClockProg（创建数字时钟）	73
96. JHCreateComplexDigClockProg（创建高级数字时钟）	75
97. JHCreateCountdownProg（倒计时节目）	79
98. JHCreateCountdownProg（正计时节目）	81
99. JHFingerProg（获取默认节目）	83
100. JHDeleteProg（删除分区节目）	85
101. JHGetBatchCnt（查询批处理文件数目）	85
102. JHGetBatchName（查询批处理文件名）	86
103. JHPreGetBatch（查询批处理文件内容第一步获取批处理文件总大小）	86
104. JHReadBatch（查询批处理文件内容第二步获取批处理文件数据）	87
105. JHFinGetBatch（查询批处理文件内容第三步结束查询，释放资源）	87
106. JHInquireBatch（查询批处理文件是否存在）	88
107. JHPreCreateBatch（添加批处理文件内容第一步发送批处理文件总大小）	88
108. JHWriteBatch（添加批处理文件内容第二步发送批处理文件数据）	89
109. JHFinCreateBatch（添加批处理文件内容第三步结束添加，释放资源）	89
110. JHDeleteBatch（删除批处理）	90
111. JHModifyBatchName（修改批处理文件名）	90
112. JHExecuteBatch(调用批处理文件)	91
113. JHEventPoweronConfig（开机事件设置）	91
114. JHEventPoweronDetail（开机事件查询）	92
115. JHEventPoweronSwitch（开机事件开关）	92
116. JHEventPoweronDelete（开机事件删除）	93
117. JHEventNocommConfig（无通讯事件设置）	93
118. JHEventNocommDetail（无通讯事件查询）	93
119. JHEventNocommSwitch（无通讯事件开关）	94
120. JHEventNocommDelete（无通讯事件删除）	94
121. JHEventTimerConfig（定时事件设置）	95
122. JHEventTimerDetail（定时事件查询）	96
123. JHEventTimerSwitch（定时事件开关）	97
124. JHEventTimerDelete（定时事件删除）	97
125. JHEventAutoPowerConfig（自动开关机配置）	97
126. JHEventAutoPowerDetail（自动开关机查询）	98
127. JHCreateBatchFile1（创建空批处理文件）	99
128. JHCreateBatchFile2（创建批处理文件）	99
129. JHDeleteBatchFile（删除批处理文件）	100
130. JHGetBatchFileData（获取批处理文件数据）	100

131. JHGetBatchFileSize (获取批处理文件大小)	101
132. JHGetBatchFileName (获取批处理文件名)	101
133. JHSetBatchFileName (设置批处理文件名)	101
134. JHSetCurLumIst (设置当前亮度指令)	102
135. JHSetCurFontIst (设置当前字体指令)	102
136. JHSetCurPenColorIst (设置当前字体颜色指令)	103
137. JHSetCurBrushColorIst (设置当前画刷颜色指令)	103
138. JHPowerIst (开关机指令)	104
139. JHRestartIst (重启指令)	104
140. JHEraseIst (清屏指令)	105
141. JHDrawMultiPixelIst (点亮点组指令)	105
142. JHDrawLineIst (画线指令)	106
143. JHFillRectIst (填充矩形指令)	107
144. JHDrawTextIst (立即显示文本指令)	107
145. JHCreateWndIst (创建分区指令)	108
146. JHChangeWndIst (改变分区属性指令)	109
147. JHDeleteWndIst (删除分区指令)	110
148. JHCreateTextProgIst (创建文本节目指令)	111
149. JHCreateSimpleDigClockProgIst (创建简易时钟指令)	114
150. JHCreateDigClockProgIst (创建数字时钟指令)	116
151. JHCreateComplexDigClockProgIst (创建高级数字时钟指令)	118
152. JHCreateBitmapProgIst (创建图片节目指令)	122
153. JHCreateQRCodeProgIst (创建二维码节目指令)	124
154. JHCreateCountDownProgIst (创建倒计时指令)	127
155. JHCreateCountUpProgIst (创建正计时指令)	129
156. JHMoveIst (移动指令)	132
157. JHDeleteIst (删除指令)	132
158. JHGetIstCnt (获取指令总数)	132
159. JHGetIstCmd (获取指令 CMD)	133
160. JHGetIstData (获取指令数据)	133
161. JHSetIstData (设置指令数据)	134
162. JHCreateGridProg (创建表格节目)	134
163. JHGridCreateHandle (创建表格实例句柄)	136
164. JHGridSetRowColCount (设置表格行数和列数)	136
165. JHGridSetDefRowHeightColWidth (设置表格默认行高和列宽)	137
166. JHGridSetDefTextFormat (设置默认表格文本格式)	137
167. JHGridSetRowHeight (设置表格指定行行高)	138
168. JHGridSetColWidth (设置表格指定列列宽)	139
169. JHGridSetCellText (设置单元格文本)	139
170. JHGridInsertRowsAt (插入行)	140
171. JHGridInsertColsAt (插入列)	140
172. JHGridRemoveColsAt (删除列)	141
173. JHGridRemoveRowsAt (删除行)	141
174. JHGridMergeCells (合并单元格)	142

175. JHGridSetCellTextFormat (设置单元格文本格式)	142
176. JHGridSetBorders (设置单元格边框).....	143
177. JHGridGetCellText (获取单元格文本内容).....	144
178. JHGridGetRowCount (获取表格行数)	145
179. JHGridGetColCount (获取表格列数).....	145
180. JHGridGetRowHeight (获取表格某一行行高).....	145
181. JHGridGetColWidth (获取表格某一列列宽).....	146
182. JHGridIsMergeCell (获取单元格是否合并信息)	146
183. JHGridGetMergeArea (获取合并区域)	147
184. JHGridGetBorders (获取单元格边框).....	147
185. JHGridGetCellTextFormat (获取单元格文本格式)	148
186. JHGridDeleteHandle (删除表格实例句柄)	149
187. JHGridFormatToHandle (表格显示格式化文本转为表格句柄)	149
188. JHGridFormatToString (表格句柄转为表格显示格式化文本).....	150
189. JHGridSetSlash (设置斜线).....	150
190. JHGridGetSlash (获取斜线).....	151
191. JHGridSetBackSlash (设置反斜线)	151
192. JHGridGetBackSlash (获取反斜线).....	152
193. JHGetBitmapCnt (查询图片数目)	152
194. JHGetBitmapId(查询图片 ID).....	153
195. JHGetBitmapInfo(查询图片信息)	153
196. JHInquireBitmap (查询图片是否存在)	154
197. JHPreCreateBitmap (下载图片数据第一步发送图片数据总大小)	154
198. JHWriteBitmap (下载图片数据第二步发送图片数据)	155
199. JHFinCreateBitmap (下载图片数据第三步结束下载, 释放资源)	156
200. JHDeleteBitmap (删除图片文件)	156
201. JHSpeechSynthesisStart (开始语音合成)	157
202. JHSpeechSynthesisStop (停止语音合成)	158
203. JHSpeechSynthesisPause (暂停语音合成)	158
204. JHSpeechSynthesisRestart (恢复语音合成)	158
205. JHSpeechSynthesisInquire (查询语音模块)	159
206. JHSpeechSynthesisSleep (休眠语音模块)	159
207. JHSpeechSynthesisWakeup (唤醒语音模块)	159
208. JHSpeechSynthesisSetPara (设置语音参数)	160
209. JHSetIOState (设置 IO 输出状态)	160
210. JHDrawBitmapIst (立即显示图片指令).....	161
211. JHSpeechSynthesisStartIst (开始语音合成指令)	162
212. JHSpeechSynthesisStopIst (停止语音合成指令)	163
213. JHSpeechSynthesisPauseIst (暂停语音合成指令)	163
214. JHSpeechSynthesisRestartIst (恢复语音合成指令)	164
215. JHSpeechSynthesisSleepIst (休眠语音模块指令)	164
216. JHSpeechSynthesisWakeupIst (唤醒语音模块指令)	164
217. JHSpeechSynthesisSetParaIst (设置语音参数指令)	165
218. JHSetIOStateIst (设置 IO 输出状态指令)	165

1. JHCreateInst（创建实例句柄）

```
JHZRESULT JHZAPI JHCreateInst(  
    HANDLE * hdl,  
    unsigned short serverid);
```

参数：

hdl	[OUT]实例句柄，指向创建好的实例。
serverid	[IN]源地址，发送端标志 ID。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_NOT_ENOUGH_CORE	内存不够

功能描述：

上位机和下位机的字库卡通讯协议中包括控制数据包传输的源地址和目的地址（相对地址（屏号）），本 API 通过指定上位机通讯协议源地址来创建一个控制卡实例，从堆中申请一块内存保存控制卡实例信息，并将实例的源地址初始化为 serverid，目的地址初始化为 0xffff。

注意：此时上位机发送的数据包目的地址为 0xffff 时，任何收到该数据包的控制卡都会处理此数据包；当用 JHChangeInstRelativeId（）修改目的地址（相对地址（屏号））为 0x0101-0xfefe 时，下位机地址和目的地址相同的处理此数据包，下位机地址和目的地址不相同的不处理此数据包。

2. JHCreateRelativeInst（创建相对实例句柄）

```
JHZRESULT JHZAPI JHCreateRelativeInst(  
    HANDLE * hdl,  
    unsigned short serverid,  
    unsigned short gid);
```

参数：

hdl	[OUT]实例句柄，指向创建好的实例。
serverid	[IN]源地址，发送端标志 ID。
gid	[IN]相对地址(屏号)，高字节为 GROUP，低字节为 ID，有效值分别为 0x01-0xFE，不能为 0x00 或 0xFF。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_NOT_ENOUGH_CORE	内存不够

功能描述：

上位机和下位机的字库卡通讯协议中包括控制数据包传输的源地址和目的地址（相

对地址（屏号）），本 API 通过指定上位机通讯协议源地址和目的地址来创建一个控制卡实例，从堆中申请一块内存保存控制卡实例信息，并将实例的源地址初始化为 `serverid`，目的地址初始化为 `gid`。

注意：此时上位机发送的数据包目的地址为 `gid`，当 `gid` 为 `0xffff` 时，此时创建的实例句柄和用 `JHCreateInst()` 创建的实例句柄相同，任何收到该数据包的控制卡都会处理此数据包，当用 `JHChangeInstRelativeId()` 修改目的地址（相对地址（屏号））为 `0x0101-0xfefe` 时，下位机地址和目的地址相同的处理此数据包，下位机地址和目的地址不相同的不处理此数据包；当 `gid` 为 `0x0101-0xfefe` 时，下位机地址和目的地址相同的处理此数据包，下位机地址和目的地址不相同的不处理此数据包。

3. JHCreateAbsoluteInst（创建绝对实例句柄）

```
JHZRESULT JHZAPI JHCreateAbsoluteInst (
    HANDLE * hdl,
    unsigned short serverid,
    const unsigned char uid[8]);
```

参数：

<code>hdl</code>	[OUT]实例句柄指针，指向创建好的实例。
<code>serverid</code>	[IN]源地址，发送端标志 ID。
<code>uid</code>	[IN]绝对地址（产品 ID），每张控制卡有一个不可修改的产品 ID。

返回值：

<code>JR_OK</code>	成功
<code>JR_INVALID_PARAMETER</code>	参数错误
<code>JR_NOT_ENOUGH_CORE</code>	内存不够

功能描述：

上位机和下位机的字库卡通讯协议中包括控制数据包传输的源地址和目的地址（相对地址（屏号）），本 API 通过指定上位机的字库卡通讯协议源地址和绝对地址（产品唯一 ID）来创建一个控制卡实例，从堆中申请一块内存保存控制卡实例信息，并将实例的源地址初始化为 `serverid`，控制卡产品唯一 `id` 初始化为 `uid`，目的地址初始化为 `0xffff`。

注意：此时上位机发送的数据包目的地址为 `0xffff` 时，任何收到该数据包的控制卡都会处理此数据包。可以通过调用 `JHGetGIDByUID()` 函数来获取目的地址（相对地址（屏号）），再通过调用 `JHChangeInstRelativeId()` 修改目的地址（相对地址（屏号））为获取的值，修改后，只有控制卡产品唯一 `id` 为 `uid` 的处理此数据包。

4. JHChangeInstServerId（修改源地址）

```
JHZRESULT JHZAPI JHChangeInstServerId(
    HANDLE hdl,
```

```
unsigned short serverid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
serverid [IN] 源地址, 发送端标志 ID。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
JR_NOT_ENOUGH_CORE 内存不够

功能描述:

通过指定上位机的字库卡通讯协议源地址来修改已经创建好的实例。该 api 设置 hdl 对应实例的源地址为 serverid。

5. JHChangeInstRelativeId(修改相对地址(屏号))

```
JHZRESULT JHZAPI JHChangeInstRelativeId(  
    HANDLE hdl,  
    unsigned short gid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
gid [IN] 相对地址(屏号), 高字节为 GROUP, 低字节为 ID, 有效值分别为 0x01-0xFE, 不能为 0x00 或 0xFF。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
JR_NOT_ENOUGH_CORE 内存不够

功能描述:

通过指定字库卡通讯协议目的地址来修改已经创建好的实例。该api设置hdl指向的实例对应的控制卡目的地址为gid, 修改后, 对应控制卡屏地址变更为gid。

6. JHChangeInstAbsoluteId(修改绝对地址 (产品 ID))

```
JHZRESULT JHZAPI JHChangeInstAbsoluteId(  
    HANDLE hdl,  
    unsigned char uid[8]);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
uid[8] [IN] 绝对地址(产品 ID), 每张控制卡有一个不可修改的产品 ID。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
JR_NOT_ENOUGH_CORE 内存不够

功能描述:

通过指定字库卡产品唯一id来修改已经创建好的实例。该api设置hdl指向的实例对应的控制卡产品唯一id为uid,修改后,实例对应的控制卡变更为产品ID为uid的控制卡,即实例对应的控制卡发生变更,而不是实例对应的控制卡的产品ID发生变更。

7. JHDeleteInst（删除实例句柄）

```
JHZRESULT JHZAPI JHDeleteInst(  
    HANDLE hdl);
```

参数:

hdl [IN] 实例句柄,与控制卡相对应。

返回值:

JR_OK 成功

功能描述:

通过指定实例句柄指针来删除已经创建好的实例。该api将删除hdl指向的实例,删除后释放保存hdl对应控制卡相关信息的内存空间。

8. JHMountTransportLayer（配置传输层）

```
JHZRESULT JHZAPI JHMountTransportLayer(  
    HANDLE hdl,  
    LPJHTRANSPORTLAYER ptl,  
    const void *argstruct);
```

参数:

hdl [IN] 实例句柄,与控制卡相对应。

ptl [IN] LPTRANSPORTLAYER指针。该指针指向的结构保存数据发送、接收等操作的函数指针,用户不需了解该结构的具体内容,只需要根据通信方式选择合适的LPTRANSPORTLAYER指针即可。具体设置请参看功能描述。

argstruct: [IN]通讯参数,具体设置请参看功能描述。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_NOT_ENOUGH_CORE	内存不够
JR_INVALID_OBJECT	无效的对象
JR_OUT_OF_RANGE	超出范围

功能描述:

为实例配置传输层,上位机通过实例与实例相对应的控制卡通讯前,都需要对该实例配置传输层。当使用串口通信时,ptl参数根据当前使用的通讯方式设置,使用串口通讯时设置为&JHTransportlayer_serialport,使用以太网帧协议通讯时设置为

&JHTransportlayer_pcap, 使用 tcp 协议通讯时设置为&JHTransportlayer_tcp, 使用 udp 协议通讯时设置&JHTransportlayer_udp, argstruct 参数也需要根据当前使用的通讯方式设置, 使用串口通讯时设置为 LPJHINFO_SERIALPORT, 使用以太网帧协议时设置为 LPJHINFO_PCAP, 使用 tcp 协议或 udp 协议时设置为 LPJHINFO_WAN。

9. JHGetGID (查询相对地址(屏号))

```
JHZRESULT JHZAPI JHGetGID (  
    HANDLE hdl,  
    unsigned short * pgid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pgid [OUT] 指向大小为 unsigned short 类型长度存储相对地址(屏号)的内存空间。相对地址(屏号)的高字节为 GROUP, 低字节为 ID, 有效值分别为 0x01-0xFE, 不能为 0x00 或 0xFF。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

读取实例句柄对应的控制卡的相对地址(屏号), 包括 GROUP 和 ID。

10. JHSetGID (配置相对地址(屏号))

```
JHZRESULT JHZAPI JHSetGID (  
    HANDLE hdl,  
    unsigned short gid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
gid [IN] 相对地址(屏号), 高字节为 GROUP, 低字节为 ID, 有效值分别为 0x01-0xFE, 不能为 0x00 或 0xFF。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

设置卡的相对地址(屏号), 屏号由组 id 成员 id 组成. 如 0x0201 02 为组 id 01 为成员 id。

11.JHGetUID（查询绝对地址（产品 ID））

```
JHZRESULT JHZAPI JHGetUID (  
    HANDLE hdl,  
    unsigned char uid[8]);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
uid[8] [OUT] 绝对地址（产品 ID），每张控制卡有一个不可修改的产品 ID。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获得实例句柄对应的控制卡的产品 id(唯一 id), 8 个字节如 0x4A01E6C39A8CE540。

12.JHGetGIDByUID(通过绝对地址（产品 ID）查询相对地址(屏号))

```
JHZRESULT JHZAPI JHGetGIDByUID (  
    HANDLE hdl,  
    unsigned char uid[8],  
    unsigned short * pgid);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
uid[8] [IN] 绝对地址（产品 ID），每张控制卡有一个不可修改的产品 ID。
pgid [OUT] 指向存储相对地址(屏号)的内存空间，该内存空间为 unsigned short 类型。相对地址(屏号)的高字节为 GROUP，低字节为 ID，有效值分别为 0x01-0xFE，不能为 0x00 或 0xFF。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

通过产品 id 获得 hdl 对应的控制卡的相对地址（屏号）。

13.JHSetGIDByUID（通过绝对地址（产品 ID）配置相对地址(屏号)）

```
JHZRESULT JHZAPI JHSetGIDByUID (  
    HANDLE hdl,  
    unsigned char uid[8],  
    unsigned short gid);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
uid[8]	[IN] 绝对地址（产品 ID），每张控制卡有一个不可修改的产品 ID。
gid	[IN] 相对地址(屏号)，高字节为 GROUP，低字节为 ID，有效值分别为 0x01-0xFE，不能为 0x00 或 0xFF。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

通过产品ID来变更实例句柄对应的控制卡的相对地址（屏号），接收到命令的控制卡首先判断自己的产品ID与命令中的uid是否相同，相同则修改自己的相对地址(屏号)。

14.JHGetSerialAttr(查询串口参数)

```
JHZRESULT JHZAPI JHGetSerialAttr(  
    HANDLE hdl,  
    unsigned long * pbaudrate,  
    unsigned char * pdatabits,  
    unsigned char * pparity,  
    unsigned char * pstopbits,  
    unsigned char * pflowctrl);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
pbaudrate	[OUT] 指向存储波特率的内存空间。
pdatabits	[OUT] 指向存储数据位的内存空间。
pparity	[OUT] 指向存储校验位的内存空间。
pstopbits	[OUT] 指向存储停止位的内存空间。
pflowctrl	[OUT] 指向存储保留位的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获得控制卡端的串口参数, 查询到的波特率可能值为 9600, 14400, 19200, 38400, 56000, 57600, 115200; 查询到的数据位可能值为 0x08 (8 位数据位), 0x09 (9 位数据位); 查询到的校验位可能值为 0x00 (无校验), 0x01 (奇校验), 0x02 (偶校验); 查询到的停止位可能值为 0xff (0.5 位停止位)。0x00 (1 位停止位), 0x01 (1.5 位停止位), 0x02 (2 位停止位); 查询到的保留为的可能值始终为 0。

15.JHSetSerialAttr (配置串口参数)

```
JHZRESULT JHZAPI JHSetSerialAttr(  
    HANDLE hdl,  
    unsigned long baudrate,  
    unsigned char databits,  
    unsigned char parity,  
    unsigned char stopbits,  
    unsigned char flowctrl);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
baudrate	[IN] 波特率, 有效值为: 9600, 14400, 19200, 38400, 56000, 57600, 115200。
databits	[IN] 数据位, 有效值为: 0x08 (8 位数据位), 0x09 (9 位数据位)。
parity	[IN] 校验位, 有效值为: 0x00 (无校验), 0x01 (奇校验), 0x02 (偶校验)。
stopbits	[IN] 停止位, 有效值为: 0xff (0.5 位停止位)。0x00 (1 位停止位), 0x01 (1.5 位停止位), 0x02 (2 位停止位)。
flowctrl	[IN] 保留位, 始终为 0。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

设置实例句柄对应控制卡的串口参数。设置生效后, 该实例需要根据串口参数的修改情况重新调用 JHMountTransportLayer 配置传输层后才能与对应控制卡正常通讯。

16.JHGetPort (查询 IP 端口号)

```
JHZRESULT JHZAPI JHGetPort(  
    HANDLE hdl,  
    unsigned short * pport)
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
-----	---------------------

pport [OUT] 指向存储 IP 端口号的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获得实例句柄对应控制卡的 TCP/IP 的端口号, 若为 0, 则表示系统使用的是默认端口 30000。

17.JHSetPort (配置 IP 端口号)

```
JHZRESULT JHZAPI JHSetPort(  
    HANDLE hdl,  
    unsigned short port)
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
port	[IN] IP 端口号。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

设置控制卡的 ip 端口号, 设置生效后, 上位机既可以使用默认端口 30000 与控制卡通讯, 又可以使用用户指定设置的端口号通讯。

18.JHGetIPv4 (查询 IPv4)

```
JHZRESULT JHZAPI JHGetIPv4(  
    HANDLE hdl,  
    unsigned long * pipctrl,  
    unsigned long * pipv4,  
    unsigned long * pnetmask,  
    unsigned long * pdefgw,  
    unsigned long * ppriids,  
    unsigned long * psecids)
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
pipctrl	[OUT] 指向存储 IP 控制参数的内存空间。
pipv4	[OUT] 指向存储控制卡 IPv4 地址的内存空间。
pnetmask	[OUT] 指向存储子网掩码的内存空间。

pdefgw	[OUT] 指向存储默认网关的内存空间。
ppridns	[OUT] 指向存储默认域名解析服务器 ip 地址的内存空间。
psecdns	[OUT] 指向存储备用域名解析服务器 ip 地址的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获得控制卡的 ipv4 信息。ip 控制参数 ipctrl 为 0 时开启 dhcp(自动获取 ip 地址), 为 1 时关闭 dhcp (使用固定 ip 地址); 控制卡 ip 地址、子网掩码、默认网关、默认域名解析服务器 ip 地址、备用域名解析服务器 ip 地址使用网络字节序存储, 如 0xC0A80103(对应点分十进制的 ip 地址 192.168.1.3)。

19.JHSetIPv4 (配置 IPv4)

```
JHZRESULT JHZAPI JHSetIPv4(
    HANDLE hdl,
    unsigned long ipctrl,
    unsigned long ipv4,
    unsigned long netmask,
    unsigned long defgw,
    unsigned long pridns,
    unsigned long secdns)
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
ipctrl	[IN] IP 控制参数。 位 31: 1, 保留, 始终为 0; 位 0, DHCP 使能, 为 0 时开启 DHCP; 为 1 时关闭 DHCP, 由上位机手动配置;
ipv4	[IN] 控制卡 IPv4 地址。
netmask	[IN] 子网掩码。
defgw	[IN] 默认网关。
pridns	[IN] 默认域名解析服务器。
secdns	[IN] 备用域名解析服务器。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

设置控制卡的 ipv4 信息。ip 控制参数 ipctrl 为 0 时开启 dhcp(自动获取 ip 地址), 为 1 时关闭 dhcp (使用固定 ip 地址); 控制卡 ip 地址、子网掩码、默认网关、默认域名解析服务器 ip 地址、备用域名解析服务器 ip 地址使用网络字节序存储, 如 0xC0A80103(对应点分十进制的 ip 地址 192.168.1.3)。

20.JHGetMac (查询 MAC)

```
JHZRESULT JHZAPI JHGetMac(  
    HANDLE hdl,  
    unsigned char mac[6])
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
mac[6] [OUT] 存储 mac 地址的数组。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获得实例句柄对应控制卡的 mac 地址。

21.JHGetLedSize(查询屏宽屏高)

```
JHZRESULT JHZAPI JHGetLedSize(  
    HANDLE hdl,  
    unsigned short * pwidth,  
    unsigned short * pheight);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pwidth [OUT] 指向存储屏宽的内存空间。
pheight [OUT] 指向存储屏高的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡的屏宽和屏高。

22.JHSetLedSize (配置屏宽屏高)

```
JHZRESULT JHZAPI JHSetLedSize(  
    HANDLE hdl,  
    unsigned short width,  
    unsigned short height);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
width [IN] 屏宽。
height [IN] 屏高。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

设置实例句柄对应控制卡的屏宽和屏高。

23.JHGetDefLedMod（查询默认单元板参数）

```
JHZRESULT JHZAPI JHGetDefLedMod(  
    HANDLE hdl,  
    JHLEDMODULE * pledmodule,  
    unsigned char * ptrack);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
pledmodule [OUT] 指向存储 led 单元板参数的内存空间。结构体 JHLEDMODULE 各个成员的含义如下：

lmScanDirection: 扫描方向
0x00: 正常;
0x01: 镜像
lmColor: 显示屏颜色
0x00: 单色;
0x01: 双色;
0x02: 彩色;
0x03: 虚拟彩色色; //保留
lmScan: 扫描; 1/x 扫描对应的 lmScan 值为 x。如静态扫描, 该值为 0x01; 1/16 扫描, 该值为 0x10;
lmOSR: 每区扫描行数;
lmOSC: 每区扫描列数;
lmRowDecode: 行译码;
0x00: 138 译码;
0x01: ABCD 直译;
lmPolar: 极性设置;
位 7: 4: 保留; 始终为 0;
位 3: 数据极性; 0 表示低电平有效, 1 表示高电平有效;
位 2: OE 极性; 0 表示低电平有效, 1 表示高电平有效;
位 1: 锁存极性; 0 表示下降沿有效, 1 表示上升沿有效;
位 0: 时钟极性; 0 表示下降沿有效, 1 表示上升沿有效;
lmClockPhase: 时钟相位, 保留, 始终为 0;
lmLatchPhase: 锁存相位; 保留, 始终为 0;

lmData: 数据组织, 如 RGB, BGR; 保留, 始终为 0;
lmReserve[6]: 保留, 始终为 0;
lmTrackMode: 走线路径(track) (方向均以正常放置的单元板的正面
为视角);

0x00	自定义
0x01	Z 型以右上角为起点
0x02	Z 型以右下角为起点
0x03	Z 型以左下角为起点
0x04	Z 型以左上角为起点
0x05	C 型以右上角为起点
0x06	C 型以右下角为起点
0x07	C 型以左下角为起点
0x08	C 型以左上角为起点
0x09	双 C 型以右上角为起点
0x0A	双 C 型以右下角为起点
0x0B	双 C 型以左下角为起点
0x0C	双 C 型以左上角为起点

lmTrackSize: 扫描路径大小; 该 API 调用成功且 lmTrackMode 为 0
时才为正数, 表示 ptrack 所占字节数; lmTrackMode 不为 0 时,
该值为 0;

ptrack [OUT] 指向存储单元板扫描路径的内存空间。仅在 API 调用成功且
lmTrackSize 不为 0 时有效。

注: 目前只有部分卡支持自定义走线模式, 对于不支持自定义走线模式的卡即 lmTrackMode
为 0, 所以传入的 lmTrackSize 应始终为 0; ptrack 指向的内存空间不需要使用, 可以传入
一个无效的地址 (如一个临时变量的地址), 但不能为 NULL。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

查询实例句柄对应控制卡的默认 (掉电不易失) 单元板参数。

24.JHSetDefLedMod (配置默认单元板参数)

```
JHZRESULT JHZAPI JHSetDefLedMod(  
    HANDLE hdl,  
    const JHLEDMODULE * pledmodule,  
    const unsigned char * ptrack);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pledmodule [IN] 指向存储 led 单元板参数的内存空间。结构体 JHLEDMODULE 各个成员的含义如下:

lmScanDirection: 扫描方向

- 0x00: 正常;
- 0x01: 镜像

lmColor: 显示屏颜色

- 0x00: 单色;
- 0x01: 双色;
- 0x02: 彩色;
- 0x03: 虚拟彩色色; //保留

lmScan: 扫描; 1/x 扫描对应的 lmScan 值为 x。如静态扫描, 该值为 0x01; 1/16 扫描, 该值为 0x10;

lmOSR: 每区扫描行数;

lmOSC: 每区扫描列数;

lmRowDecode: 行译码;

- 0x00: 138 译码;
- 0x01: ABCD 直译;

lmPolar: 极性设置;

- 位 7: 4: 保留; 始终为 0;
- 位 3: 数据极性; 0 表示低电平有效, 1 表示高电平有效;
- 位 2: OE 极性; 0 表示低电平有效, 1 表示高电平有效;
- 位 1: 锁存极性; 0 表示下降沿有效, 1 表示上升沿有效;
- 位 0: 时钟极性; 0 表示下降沿有效, 1 表示上升沿有效;

lmClockPhase: 时钟相位, 保留, 始终为 0;

lmLatchPhase: 锁存相位; 保留, 始终为 0;

lmData: 数据组织, 如 RGB, BGR; 保留, 始终为 0;

lmReserve[6]: 保留, 始终为 0;

lmTrackMode: 走线路径(track) (方向均以正常放置的单元板的正面为视角);

0x00	自定义
0x01	Z 型以右上角为起点
0x02	Z 型以右下角为起点
0x03	Z 型以左下角为起点
0x04	Z 型以左上角为起点
0x05	C 型以右上角为起点
0x06	C 型以右下角为起点
0x07	C 型以左下角为起点
0x08	C 型以左上角为起点
0x09	双 C 型以右上角为起点
0x0A	双 C 型以右下角为起点
0x0B	双 C 型以左下角为起点
0x0C	双 C 型以左上角为起点

lmTrackSize: 扫描路径大小; 该 API 调用成功且 lmTrackMode 为 0 时才为正数, 表示 ptrack 所占字节数; lmTrackMode 不为 0 时, 该值为 0;

ptrack [IN] 单元板扫描路径指针。

注：目前只有部分卡支持自定义走线模式，对于不支持自定义走线模式的卡即 lmTrackMode 为 0，所以传入的 lmTrackSize 应始终为 0；ptrack 指向的内存空间不需要使用，可以传入一个无效的地址（如一个临时变量的地址），但不能为 NULL。

返回值：

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能描述：

配置实例句柄对应控制卡的默认（掉电不易失）单元板参数。

25.JHGetCurLedMod（查询当前单元板参数）

```
JHZRESULT JHZAPI JHGetCurLedMod(  
    HANDLE hdl,  
    JHLEDMODULE* pledmodule,  
    unsigned char * ptrack);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

pledmodule [OUT] 指向存储 led 单元板参数的内存空间。结构体 JHLEDMODULE 各个成员的含义如下：

lmScanDirection: 扫描方向

0x00: 正常;

0x01: 镜像

lmColor: 显示屏颜色

0x00: 单色;

0x01: 双色;

0x02: 彩色;

0x03: 虚拟彩色色; //保留

lmScan: 扫描; 1/x 扫描对应的 lmScan 值为 x。如静态扫描, 该值为

0x01; 1/16 扫描, 该值为 0x10;

lmOSR: 每区扫描行数;

lmOSC: 每区扫描列数;

lmRowDecode: 行译码;

0x00: 138 译码;

0x01: ABCD 直译;

lmPolar: 极性设置;

位 7: 4: 保留; 始终为 0;

位 3: 数据极性; 0 表示低电平有效, 1 表示高电平有效;

位 2: OE 极性; 0 表示低电平有效, 1 表示高电平有效;

位 1: 锁存极性; 0 表示下降沿有效, 1 表示上升沿有效;

位 0: 时钟极性; 0 表示下降沿有效, 1 表示上升沿有效;

lmClockPhase: 时钟相位, 保留, 始终为 0;

lmLatchPhase: 锁存相位; 保留, 始终为 0;

lmData: 数据组织, 如 RGB, BGR; 保留, 始终为 0;
lmReserve[6]: 保留, 始终为 0;
lmTrackMode: 走线路径(track) (方向均以正常放置的单元板的正面
为视角);

0x00	自定义
0x01	Z 型以右上角为起点
0x02	Z 型以右下角为起点
0x03	Z 型以左下角为起点
0x04	Z 型以左上角为起点
0x05	C 型以右上角为起点
0x06	C 型以右下角为起点
0x07	C 型以左下角为起点
0x08	C 型以左上角为起点
0x09	双 C 型以右上角为起点
0x0A	双 C 型以右下角为起点
0x0B	双 C 型以左下角为起点
0x0C	双 C 型以左上角为起点

lmTrackSize: 扫描路径大小; 该 API 调用成功且 lmTrackMode 为 0
时才为正数, 表示 ptrack 所占字节数; lmTrackMode 不为 0 时,
该值为 0;

ptrack [OUT] 指向存储单元板扫描路径的内存空间。仅在 API 调用成功且
lmTrackSize 不为 0 时有效。

注: 目前不支持 lmTrackMode 为 0 即自定义走线模式, 所以查询到的 lmTrackSize 值为 0,
ptrack 指向的内存空间不需要使用, 可以传入一个无效的地址 (如一个临时变量的地址),
但不能为 NULL。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

查询实例句柄对应控制卡的当前 (掉电易失) 单元板参数。

26.JHSetCurLedMod (配置当前单元板参数)

```
JHZRESULT JHZAPI JHSetCurLedMod(  
    HANDLE hdl,  
    const JHLEDMODULE* pledmodule,  
    const unsigned char * ptrack);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pledmodule [IN] 指向存储 led 单元板参数的内存空间。结构体 JHLEDMODULE 各个成员的含义如下:

lmScanDirection: 扫描方向

- 0x00: 正常;
- 0x01: 镜像

lmColor: 显示屏颜色

- 0x00: 单色;
- 0x01: 双色;
- 0x02: 彩色;
- 0x03: 虚拟彩色色; //保留

lmScan: 扫描; 1/x 扫描对应的 lmScan 值为 x。如静态扫描, 该值为 0x01; 1/16 扫描, 该值为 0x10;

lmOSR: 每区扫描行数;

lmOSC: 每区扫描列数;

lmRowDecode: 行译码;

- 0x00: 138 译码;
- 0x01: ABCD 直译;

lmPolar: 极性设置;

- 位 7: 4: 保留; 始终为 0;
- 位 3: 数据极性; 0 表示低电平有效, 1 表示高电平有效;
- 位 2: OE 极性; 0 表示低电平有效, 1 表示高电平有效;
- 位 1: 锁存极性; 0 表示下降沿有效, 1 表示上升沿有效;
- 位 0: 时钟极性; 0 表示下降沿有效, 1 表示上升沿有效;

lmClockPhase: 时钟相位, 保留, 始终为 0;

lmLatchPhase: 锁存相位; 保留, 始终为 0;

lmData: 数据组织, 如 RGB, BGR; 保留, 始终为 0;

lmReserve[6]: 保留, 始终为 0;

lmTrackMode: 走线路径(track) (方向均以正常放置的单元板的正面为视角);

0x00	自定义
0x01	Z 型以右上角为起点
0x02	Z 型以右下角为起点
0x03	Z 型以左下角为起点
0x04	Z 型以左上角为起点
0x05	C 型以右上角为起点
0x06	C 型以右下角为起点
0x07	C 型以左下角为起点
0x08	C 型以左上角为起点
0x09	双 C 型以右上角为起点
0x0A	双 C 型以右下角为起点
0x0B	双 C 型以左下角为起点
0x0C	双 C 型以左上角为起点

lmTrackSize: 扫描路径大小; 该 API 调用成功且 lmTrackMode 为 0 时才为正数, 表示 ptrack 所占字节数; lmTrackMode 不为 0 时, 该值为 0;

ptrack [IN] 单元板扫描路径指针。

注：目前不支持 lmTrackMode 为 0 即自定义走线模式，所以传入的 lmTrackSize 应始终为 0；ptrack 指向的内存空间不需要使用，可以传入一个无效的地址（如一个临时变量的地址），但不能为 NULL。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

配置实例句柄对应控制卡的默认（掉电易失）单元板参数。

27.JHGetDefRefreq（查询扫描频率）

```
JHZRESULT JHZAPI JHGetDefRefreq(  
    HANDLE hdl,  
    unsigned short * prefreq);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
prefreq	[OUT] 指向存储扫描频率的内存空间，扫描频率的有效值为 0x0000-0x0190。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡的扫描频率。

28.JHSetDefRefreq(配置扫描频率)

```
JHZRESULT JHZAPI JHSetDefRefreq(  
    HANDLE hdl,  
    unsigned short refreq);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
refreq	[IN] 显示屏扫描频率，有效值为 0x0000-0x0190。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置实例句柄对应控制卡的扫描频率。

29.JHGetDefLum（查询默认亮度）

```
JHZRESULT JHZAPI JHGetDefLum(  
    HANDLE hdl,  
    unsigned short * plum);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
plum [OUT] 指向存储亮度的内存空间, 亮度的有效值为 0x0000-0x03DE。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

查询实例句柄对应控制卡的默认（掉电不易失）亮度值。

30.JHSetDefLum（配置默认亮度）

```
JHZRESULT JHZAPI JHSetDefLum(  
    HANDLE hdl,  
    unsigned short lum);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
lum [IN] 亮度, 有效值为 0x0000-0x03DE。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

配置实例句柄对应控制卡的默认（掉电不易失）亮度值。

31.JHGetCurLum（查询当前亮度）

```
JHZRESULT JHZAPI JHGetCurLum(  
    HANDLE hdl,  
    unsigned short * plum);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
plum [OUT] 指向存储亮度的内存空间, 亮度的有效值为 0x0000-0x03DE。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

查询实例句柄对应控制卡当前（掉电易失）显示屏亮度值。

32.JHSetCurLum（配置当前亮度）

```
JHZRESULT JHZAPI JHSetCurLum(
    HANDLE hdl,
    unsigned short lum);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

lum [IN] 亮度，有效值为 0x0000-0x03DE。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置显示屏当前（掉电易失）亮度。

33.JHGetDefFont(查询默认字库文件)

```
JHZRESULT JHZAPI JHGetDefFont(
    HANDLE hdl,
    unsigned char * pfid);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

pfid [OUT]指向存储字体 ID 的内存空间，字体 ID 的有效值为 1-254。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

查询默认（掉电不易失）字体 ID；字体 ID 决定了字体和字体大小；字体 ID 在生成字库是指定。

34.JHSetDefFont(配置默认字库文件)

```
JHZRESULT JHZAPI JHSetDefFont(  
    HANDLE hdl,  
    unsigned char fid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
fid [IN] 字体 ID, 有效值为 1-254。

返回值:

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述:

配置默认(掉电不易失)字库, 如 id 为 1 表示把字体 id 为 1 的字库设为默认的, 控制卡可以存在多个字库, 但默认字库只能存在一个。

35.JHGetCurFont(查询当前字库文件)

```
JHZRESULT JHZAPI JHGetCurFont(  
    HANDLE hdl,  
    unsigned char * pfid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pfid [OUT] 指向存储字体 ID 的内存空间, 字体 ID 的有效值为 1-254。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
JR_INVALID_OBJECT 无效的对象

功能描述:

获取实例句柄对应控制卡的当前(掉电易失)字库文件的字体 ID。

36.JHSetCurFont (配置当前字库文件)

```
JHZRESULT JHZAPI JHSetCurFont(  
    HANDLE hdl,  
    unsigned char fid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
fid [IN] 字体 ID, 有效值为 1-254。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置实例句柄对应控制卡当前（掉电易失）字库文件的字体ID。

37.JHGetDefPenColor（查询默认画笔颜色）

```
JHZRESULT JHZAPI JHGetDefPenColor(  
    HANDLE hdl,  
    unsigned long * pcolor);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pcolor	[OUT] 指向存储画笔颜色的内存空间，画笔颜色即为字体颜色，有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡默认（掉电不易失）的画笔颜色，画笔颜色即为字体颜色。

38.JHSetDefPenColor（配置默认画笔颜色）

```
JHZRESULT JHZAPI JHSetDefPenColor(  
    HANDLE hdl,  
    unsigned long color);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
color	[IN] 画笔颜色，有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置实例句柄对应控制卡默认（掉电不易失）的画笔颜色，画笔颜色即为字体颜色。

39.JHGetCurPenColor（查询当前画笔颜色）

```
JHZRESULT JHZAPI JHGetCurPenColor(  
    HANDLE hdl,  
    unsigned long * pcolor);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pcolor	[OUT] 指向存储画笔颜色的内存空间，画笔颜色即为字体颜色，有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡当前（掉电易失）画笔颜色，画笔颜色即为字体颜色。

40.JHSetCurPenColor（配置当前画笔颜色）

```
JHZRESULT JHZAPI JHSetCurPenColor(  
    HANDLE hdl,  
    unsigned long color);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
color	[IN] 画笔颜色，有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红；

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置控制卡当前（掉电易失）使用的画笔颜色，画笔颜色即为字体颜色。

41.JHGetDefBrushColor（查询默认画刷颜色）

```
JHZRESULT JHZAPI JHGetDefBrushColor(  
    HANDLE hdl,  
    unsigned long * pcolor);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pcolor	[OUT] 指向存储画刷颜色的内存空间，画刷颜色的有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡默认（掉电不易失）画刷颜色。

42.JHSetDefBrushColor（配置默认画刷颜色）

```
JHZRESULT JHZAPI JHSetDefBrushColor(  
    HANDLE hdl,  
    unsigned long color);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
color	[IN] 画刷颜色，有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红；

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置实例句柄对应控制卡默认（掉电不易失）画刷颜色。

43.JHGetCurBrushColor（查询当前画刷颜色）

```
JHZRESULT JHZAPI JHGetCurBrushColor(  
    HANDLE hdl,  
    unsigned long * pcolor);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pcolor	[OUT] 指向画刷颜色的内存空间，画刷的有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡当前（掉电易失）画刷颜色。

44.JHSetCurBrushColor（配置当前画刷颜色）

```
JHZRESULT JHZAPI JHSetCurBrushColor(  
    HANDLE hdl,  
    unsigned long color);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
color	[IN] 画刷颜色，有效值为： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝； 位 1：绿； 位 0：红；

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

配置实例句柄对应控制卡当前（掉电易失）画刷颜色。

45.JHGetLocale(查询区域语言)

```
JHZRESULT JHZAPI JHGetLocale(  
    HANDLE hdl,  
    unsigned short * plocale);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
plocale	[OUT]指向存储区域语言的内存空间，区域语言有效值为： 英文：437； 简体中文：936； 繁体中文：950；

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡的区域语言信息，区域语言即为系统所使用语言。

46.JHGetSystemTime（查询系统时间）

```
JHZRESULT JHZAPI JHGetSystemTime(  
    HANDLE hdl,  
    JHSYSTEMTIME * ptime);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
ptime	[OUT]指向存储系统时间的内存空间，系统时间数据结构如下所示： wYear: 年； wMonth: 月，有效值为 1-12； wDayOfWeek: 星期，有效值为 0-6； 星期天: 0 星期一: 1 星期二: 2 星期三: 3

星期四: 4

星期五: 5

星期六: 6

wDay: 日, 有效值为 1-31;

wHour: 时, 有效值为 0-23;

wMinute: 分, 有效值为 0-59;

wSecond: 秒, 有效值为 0-59;

wMilliseconds: 毫秒, 保留值, 始终为 0;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡的系统时间。

47.JHSetSystemTime（配置系统时间）

```
JHZRESULT JHZAPI JHSetSystemTime(  
    HANDLE hdl,  
    const JHSYSTEMTIME * ptime);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
ptime	[IN] 指向存储系统时间的内存空间, 系统时间数据结构如下所示: wYear: 年; wMonth: 月, 有效值为 1-12; wDayOfWeek: 星期, 有效值为 0-6; 星期天: 0 星期一: 1 星期二: 2 星期三: 3 星期四: 4 星期五: 5 星期六: 6 wDay: 日, 有效值为 1-31; wHour: 时, 有效值为 0-23; wMinute: 分, 有效值为 0-59; wSecond: 秒, 有效值为 0-59; wMilliseconds: 毫秒, 保留值, 始终为 0;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

配置实例句柄对应控制卡的系统时间。

48.JHGetSystemVersion（查询系统版本号）

```
JHZRESULT JHZAPI JHGetSystemVersion(  
    HANDLE hdl,  
    unsigned long * phwver,  
    unsigned long * pswver);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
phwver [OUT] 指向储存硬件版本号的内存空间。
pswver [OUT] 指向存储软件版本号的内存空间。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡的系统的版本信息，包括硬件版本和软件版本。

49.JHGetRestCapacity（查询 flash 剩余空间）

```
JHZRESULT JHZAPI JHGetRestCapacity(  
    HANDLE hdl,  
    unsigned long * pcapacity);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
pcapacity [OUT] 指向存储 flash 剩余空间大小的内存空间，单位：字节。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡 flash 剩余可用空间大小，单位：字节。

50.JHGetTotalCapacity（查询 flash 空间总大小）

```
JHZRESULT JHZAPI JHGetTotalCapacity(  
    HANDLE hdl,
```

```
HANDLE hdl,  
unsigned long * pcapacity);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pcapacity [OUT] 指向存储 flash 空间总大小的内存空间, 单位: 字节。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡的 flash 总容量, 单位: 字节。

51.JHGetSilentMode (查询静默模式)

```
JHZRESULT JHZAPI JHGetSilentMode(  
HANDLE hdl,  
unsigned char * psilentmode);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
psilentmode [OUT] 指向存储静默模式状态的内存空间, 获取的值为 0x00 时表示关闭静默状态, 为 0x01 时表示开启静默状态。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获得实例句柄对应控制卡的静默模式状态, 0x00 表示关闭状态, 0x01 表示开启状态。
开后静默状态后, 控制卡处理完上位机的命令后不回复上位机。

52.JHSetSilentMode (配置静默模式)

```
JHZRESULT JHZAPI JHSetSilentMode(  
HANDLE hdl,  
unsigned char silentmode);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
silentmode [IN] 静默模式状态, 0x00 表示关闭状态, 0x01 表示开启状态。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

配置实例句柄对应控制卡的静默模式状态，0x00 表示关闭状态，0x01 表示开启状态。开后静默状态后，控制卡处理完上位机的命令后不回复上位机。

53.JHPower（开关机）

```
JHZRESULT JHZAPI JHPower(  
    HANDLE hdl,  
    unsigned char onoff);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
onoff [IN] 开关机控制状态，0x00 表示关机，其他表示开机。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

适用于手动控制实例句柄对应的控制卡开机或关机。onoff 等于 0x00 表示关机，其他表示开机。

54.JHRestart（重启）

```
JHZRESULT JHZAPI JHRestart(  
    HANDLE hdl);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

适用于重新启动实例句柄对应的控制卡。

55.JHPreUpgrade（升级第一步发送升级数据总大小）

```
JHZRESULT JHZAPI JHPreUpgrade(  
    HANDLE hdl,  
    HANDLE *fd,  
    unsigned long writesize);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

fd [OUT] 指向升级第一步中申请的存储升级过程所需信息的内存空间，升级第二步需要根据这个信息进行操作，第三步会释放 fd 指向的内存空间。

writesize [IN] 升级数据总大小, 单位：字节。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不够

功能描述：

升级操作的第一步。申请一块内存空间存储升级过程中所需要的信息，提供给升级过程第二步操作时使用。

使用示例：

```
HANDLE hd;
JHPreUpgrade(hdl, &hd, writesize);
JHWriteUpgrade(hdl, fd, woffset, pdat, size); //文件较大可分多次发送
JHFinUpgrade(hdl, fd);
```

56.JHWriteUpgrade（升级第二步发送升级数据）

```
JHZRESULT JHZAPI JHWriteUpgrade(
    HANDLE hdl,
    HANDLE fd,
    unsigned long woffset,
    const char * pdat,
    unsigned long size);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

fd [IN] fd 指向升级第一步中申请的存储升级过程所需信息的内存空间，升级第二步需要根据这个信息进行操作，第三步会释放 fd 指向的内存空间。

woffset [IN] 本次所要发送数据在升级数据中的偏移，等于上一个数据包 Woffset + 上一个数据包 size 大小。

pdat [IN] 指向本次所要发送数据的指针。

size [IN] 本次所要发送数据的大小, 不超过 1200 字节，不少于 1 个字节。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

升级操作的第二步，发送升级数据到控制卡上，当升级数据较大时，需分包发送。

使用示例：

```
HANDLE hd;
char[4000] dat={1};
```

```

    int num;
    JHPreUpgrade(hdl, &hd, size);
    pdat = dat;
    num = 0;
    if(size > 1200)
    {
        While(num < size)
        {
            JHWriteUpgrade(hdl, fd, pdat - dat, pdat, num+1200<size ? 1200 : size-num);
            num+=1200;
        }
    }
    JHFinUpgrade(hdl, fd);

```

57.JHFinUpgrade（升级第三步结束升级，释放资源）

```

JHZRESULT JHZAPI JHFinUpgrade(
    HANDLE hdl,
    HANDLE fd);

```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
 fd [IN] fd 指向升级第一步中申请的存储升级过程所需信息的内存空间，升级第二步需要根据这个信息进行操作，第三步会释放 fd 指向的内存空间。

返回值:

JR_OK 成功
 JR_INVALID_OBJECT 无效的对象

功能描述:

通知实例句柄对应控制卡升级数据发送完成，释放fd资源。

58.JHRestoreFactory（恢复参数出厂设置）

```

JHZRESULT JHZAPI JHRestoreFactory(
    HANDLE hdl);

```

参数:

hdl [IN] 实例句柄，与控制卡相对应。

返回值:

JR_OK 成功
 JR_INVALID_OBJECT 无效的对象

功能描述:

用于将系统各项参数和配置恢复到出厂状态，需要注意恢复出厂设置后通讯方式的

变化。

59.JHRestoreSysFactory（恢复系统出厂设置）

```
JHZRESULT JHZAPI JHRestoreSysFactory(  
    HANDLE hdl);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

用于将整个系统恢复到出厂状态，需要注意恢复出厂设置后通讯方式的变化。

注：以当前状态回应，回应后需等待约 1 分钟左右生效，或者如果控制卡接屏的话屏会亮一下说明已经恢复生效。

60.JHGetFontCnt（查询字体数目）

```
JHZRESULT JHZAPI JHGetFontCnt(  
    HANDLE hdl,  
    unsigned char * pcnt);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

pcnt [OUT] 指向存储字体数目的内存空间。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡里字库的数目。

61.JHGetFontId(查询字体 ID)

```
JHZRESULT JHZAPI JHGetFontId(  
    HANDLE hdl,  
    unsigned char index,  
    unsigned char * pfid);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 字体序号；系统中该序号为连续序号，区分不同的字体；该值从 0 开始。
pfid [OUT] 指向存储字体 ID 的内存空间，字体 ID 的有效值为 1-254，用于标识字库文件。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡中指定序号的字体 ID，序号从 0 开始，字体 ID 的有效值 1-254。

62.JHGetFontInfo(查询字体详细信息)

```
JHZRESULT JHZAPI JHGetFontInfo(  
    HANDLE hdl,  
    unsigned char fid,  
    JHLOGFONT * plogfont);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
fid	[IN] 字体 ID，有效值为 1-254，用于标识字库文件。
plogfont	[OUT] 指向 JHLOGFONT 类型的内存空间，JHLOGFONT 结构体如下所示： lfHeight：字体高度； lfWidth：字体宽度； lfEscapement：指定每一行文本输出时相对于页面底端的角度； lfOrientation：字符基线相对于页面底端的角度； lfWeight：字体重量； lfItalic：为 TRUE 时使用斜体； lfUnderline：为 TRUE 时给字体添加下划线； lfStrikeOut：为 TRUE 时给字体添加删除线； lfCharSet：指定字符集； lfOutPrecision：指定输出精度； lfClipPrecision：指定剪辑精度； lfQuality：定义输出质量； lfPitchAndFamily：指定字体的字符间距和族 lfFaceName[32]：以 NULL 结尾的字符串，它指定所用的字体名；

注：关于 JHLOGFONT 的详细信息请参考 win 下的 LOGFONT 结构。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡中字体 ID 号为 fid 的字库详细信息。

63.JHGetFontInfoEx(高级查询字体详细信息)

```
JHZRESULT JHZAPI JHGetFontInfoEx(  
    HANDLE hdl,  
    unsigned char fid,  
    unsigned short * pver,  
    void * pBuf,  
    unsigned short * psize)
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
fid	[IN] 字体 ID, 有效值为 1-254, 用于标识字库文件。
pver	[OUT] 指向存储字库文件版本号的内存空间。
pBuf	[OUT] 指向存储字体详细信息的内存空间, 由用户申请, 大小为 256 字节。
psize	[IN][OUT]: 调用 API 时需初始化为 pBuf 的总大小, 返回成功时存储实际读取数据的长度。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

根据字体 ID, 获得字体高级信息, pver 字库文件的版本号。

注: 本 API 暂不使用。

64.JHInquireFont (查询字体是否存在)

```
JHZRESULT JHZAPI JHInquireFont(  
    HANDLE hdl,  
    unsigned char fid);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
fid	[IN] 字体 ID, 有效值为 1-254, 用于标识字库文件。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

查询实例句柄对应控制卡中字体 ID 为 fid 的字库是否存在, 返回 JR_OK 表示存在, 否则为不存在。

65.JHPreCreateFont（下载字库数据第一步发送字库数据总大小）

```
JHZRESULT JHZAPI JHPreCreateFont(  
    HANDLE hdl,  
    HANDLE *fd,  
    unsigned char fid,  
    unsigned long writesize);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
fd	[OUT] 指向发送字库第一步中申请的存储字库下载操作过程信息的内存空间，发送字库第二步需要根据这个信息进行操作，第三步会释放 fd 指向的内存空间。
fid	[IN] 字体 ID，有效值为 1-254，用于标识字库文件。
writesize	[IN] 字库数据总大小。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不够

功能描述：

发送字库数据操作的第一步，申请一块内存保存下载操作过程的信息，提供给发送字库操作第二步使用。

66.JHWriteFont（下载字库数据第二步发送字库数据）

```
JHZRESULT JHZAPI JHWriteFont(  
    HANDLE hdl,  
    HANDLE fd,  
    unsigned long woffset,  
    const char * pdat,  
    unsigned long size);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
fd	[IN] 指向发送字库第一步中申请的存储字库下载操作过程信息的内存空间，发送字库第二步需要根据这个信息进行操作，第三步会释放 fd 指向的内存空间。
woffset	[IN] 本次所要发送数据在字库数据中的偏移。
pdat	[IN] 指向存储本次所要发送数据的内存空间。
size	[IN] 本次所要发送数据的大小，不超过 1200 字节，不少于 1 个字节。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

发送字库数据到实例句柄对应的控制卡上, 当字库内容较大时, 需分包发送。

使用示例:

```
HANDLE hd;  
char[4000] dat={1};  
int num, fid;  
fid = 1;  
JHPreCreateFont (hdl, &hd, fid, size);  
pdat = dat;  
num = 0;  
if(size > 1200)  
{  
    while(num < size)  
    {  
        JHWriteFont(hdl, fd, pdat - dat, pdat, num+1200<size? 1200: size-num);  
        num+=1200;  
    }  
}  
JHFinCreateFont(hdl, fd);
```

67.JHFinCreateFont（下载字库数据第三步结束下载，释放资源）

```
JHZRESULT JHZAPI JHFinCreateFont(  
    HANDLE hdl,  
    HANDLE fd);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
fd	[IN] 指向发送字库第一步中申请的存储字库下载操作过程信息的内存空间, 发送字库第二步需要根据这个信息进行操作, 第三步会释放 fd 指向的内存空间。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

通知实例句柄对应控制卡字库数据发送完成, 释放fd资源。

68.JHDeleteFont（删除字库文件）

```
JHZRESULT JHZAPI JHDeleteFont(  
    HANDLE hdl,  
    unsigned char fid);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
fid [IN] 字体 ID，有效值为 1-254，用于标识字库文件。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

删除实例句柄对应控制卡中字体 ID 为 fid 的字库文件。

69.JHErase(清屏)

```
JHZRESULT JHZAPI JHErase(  
    HANDLE hdl);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

清除实例句柄对应控制卡的整个屏幕的显示内容。

70.JHDrawMultiPixel（点亮点组）

```
JHZRESULT JHZAPI JHDrawMultiPixel(  
    HANDLE hdl,  
    const char * pdat,  
    unsigned short poscnt);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
pdat [IN] 指向存储点组数据的内存空间，每个点数据包括 x 坐标和 y 坐标，格式为：(X1 Y1 X2 Y2 ... Xn Yn)，其中 Xn 和 Yn 均为 2 字节。
poscnt [IN] 点组总数。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡上以当前画笔颜色（即当前字体颜色）点亮 pdat 确定的点组。

71.JHDrawMultiPixelEx（高级点亮点组）

```
JHZRESULT JHZAPI JHDrawMultiPixelEx(
    HANDLE hdl,
    const char * pdat,
    unsigned short poscnt,
    unsigned long color);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

pdat [IN] 指向存储点组数据的内存空间，每个点数据包括 x 坐标和 y 坐标，格式为：(X1 Y1 X2 Y2 ... Xn Yn)，其中 Xn 和 Yn 均为 2 字节。

poscnt [IN] 点组总数。

color [IN] 点组颜色，格式为：

- 位 7-4：保留，始终为 0；
- 位 3-0：颜色（注：不能为 0000；黑色为 1000）
- 位 2：蓝；
- 位 1：绿；
- 位 0：红；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡上以color参数确定的颜色点亮pdat确定的点组。

72.JHDrawLine（画线）

```
JHZRESULT JHZAPI JHDrawLine(
    HANDLE hdl,
    short x1,
    short y1,
    short x2,
    short y2);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
x1 [IN] 直线起始点 X 坐标。
y1 [IN] 直线起始点 Y 坐标。
x2 [IN] 直线结束点 X 坐标。
y2 [IN] 直线结束点 Y 坐标。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

在实例句柄对应控制卡上以当前画笔颜色（即当前字体颜色）画出起点坐标为（x1, y1），终点坐标为（x2, y2）的直线。

73.JHDrawLineEx（高级画线）

```
JHZRESULT JHZAPI JHDrawLineEx(  
    HANDLE hdl,  
    short x1,  
    short y1,  
    short x2,  
    short y2,  
    unsigned long color);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
x1 [IN] 直线起始点 X 坐标。
y1 [IN] 直线起始点 Y 坐标。
x2 [IN] 直线结束点 X 坐标。
y2 [IN] 直线结束点 Y 坐标。
color [IN] 直线颜色，格式为：
 位 7-4：保留，始终为 0；
 位 3-0：颜色（注：不能为 0000；黑色为 1000）
 位 2：蓝；
 位 1：绿；
 位 0：红；

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

在实例句柄对应控制卡上以 color 参数确定的颜色画出起始点坐标为（x1, y1），终点坐标为（x2, y2）的直线。

74.JHFrameRect（画矩形）

```
JHZRESULT JHZAPI JHFrameRect(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
x	[IN] 矩形左上角起始点 X 坐标。
y	[IN] 矩形左上角起始点 Y 坐标。
width	[IN] 矩形宽度。
height	[IN] 矩形高度。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡上根据给定的左上角起点坐标和宽、高，用当前画笔颜色（即当前字体颜色）在显示屏上画一个矩形边框。

75.JHFrameRectEx（高级画矩形）

```
JHZRESULT JHZAPI JHFrameRectEx(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height,  
    unsigned long color);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
x	[IN] 矩形左上角起始点 X 坐标。
y	[IN] 矩形左上角起始点 Y 坐标。
width	[IN] 矩形宽度。
height	[IN] 矩形高度。
color	[IN] 矩形边框颜色，格式如下： 位 7-4：保留，始终为 0； 位 3-0：颜色（注：不能为 0000；黑色为 1000） 位 2：蓝；

位 1: 绿;

位 0: 红;

返回值:

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述:

在实例句柄对应控制卡上根据给定的左上角起点坐标和宽、高, 用 color 确定的画笔颜色在屏上画一个矩形边框。

76.JHFillRect (填充矩形)

```
JHZRESULT JHZAPI JHFillRect(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

x [IN] 矩形左上角起始点 X 坐标。

y [IN] 矩形左上角起始点 Y 坐标。

width [IN] 矩形宽度。

height [IN] 矩形高度。

返回值:

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述:

在实例句柄对应控制卡上根据给定的左上角起点坐标和宽、高, 用当前画刷颜色在屏上填充一个矩形。

77.JHFillRectEx (高级填充矩形)

```
JHZRESULT JHZAPI JHFillRectEx(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height,  
    unsigned long color);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。

x [IN] 矩形起始点 X 坐标。

y [IN] 矩形起始点 Y 坐标。

width [IN] 矩形宽度。

height [IN] 矩形高度。

color [IN] 矩形颜色，格式为：

位 7-4：保留，始终为 0；

位 3-0：颜色（注：不能为 0000；黑色为 1000）

位 2：蓝；

位 1：绿；

位 0：红；

返回值：

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述：

在实例句柄对应控制卡上根据给定的左上角起点坐标和宽、高，用 color 确定的画刷颜色在屏上填充一个矩形。

78.JHDrawText（立即显示文本）

```
JHZRESULT JHZAPI JHDrawText(
    HANDLE hdl,
    short x,
    short y,
    unsigned short width,
    unsigned short height,
    unsigned long format,
    const char * pformatstring);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

x [IN] 文本显示位置左上角的 X 坐标。

y [IN] 文本显示位置左上角的 Y 坐标。

width [IN] 文本显示区域宽度，为 0x0000 时为自适应。

height [IN] 文本显示区域高度，为 0x0000 时为自适应。

format [IN] 文本显示格式，如下所示：

位 31-20：保留，始终为 0；

位 19- 16：颜色（注：0000 表示使用系统当前颜色；1000 为黑色）

位 18：蓝色；

位 17：绿色；

位 16：红色；

位 15- 8：字体 ID；有效值为 0-254；为 0 时使用当前字体 ID；

位 7- 6: 水平对齐 HALIGN
 00: 左对齐;
 01: 水平居中对齐;
 10: 右对齐;
 位 5- 4: 垂直对应方式
 00: 上对齐;
 01: 垂直居中对齐;
 10: 下对齐;
 位 3: 保留, 始终为 0;
 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本;
 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0;

pformatstring [IN] 指向存储待显示的文本字符串的内存空间 (1-128 字节)。文本字符编码, 可 包含有效的 ASCII 码 (不能包含 0x00) 和 GB2312 编码, 可混用。换行符为 ' \n' , 即 0x0A; 水平制表符 ' \t' , 即 0x09, 显示为 4 个 ASCII 空格符。

(注: FORMAT 属性说明, 主要注意两类, 单行文本和多行文本

1、单行文本时:

忽略 wordbreak 自动换行设置, 无论是自动换行或是手动在命令中添加换行符 ' \n' , 均忽略, 不作换行处理,

2、多行文本时:

根据设置不作限制, 自动换行和手动换行可同时支持;

但是若文本显示所需区域超过当前限制区域, 建议做如下处理:

1、若为单行文本, 建议不要设置水平居中; 其他可任意搭配;

2、若为多行文本, 若使用了自动换行, 建议不要使用垂直居中对齐; 若无自动换行, 建议不要使用任何居中对齐方式;

使用时若遇到上面情况, 可参照上面说明使用, 否则为了整体显示效果, 系统会自行裁剪部分数据;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应控制卡上在 x, y, width, height 确定的矩形区域内按照 format 参数确定的格式显示文本, 文本内容由 pformatstring 指定。

79.JHDrawBitmap (立即显示图片)

```
JHZRESULT JHZAPI JHDrawBitmap(
    HANDLE hdl,
```

```

short x,
short y,
unsigned short width,
unsigned short height,
unsigned long format,
const JHBMP_SRC * pbmpsrc);

```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
x	[IN] 图片显示位置左上角的 X 坐标。
y	[IN] 图片显示位置左上角的 Y 坐标。
width	[IN] 图片显示区域宽度。
height	[IN] 图片显示区域高度。
Format	[IN] 图片显示格式, 如下所示: 位 31-8: 保留, 为 0; 位 7- 6: 水平对齐 HALIGN 00: 左对齐; 01: 水平居中对齐; 10: 右对齐; 位 5- 4: 00: 上对齐; 01: 垂直居中对齐; 10: 下对齐; 位 3-0: 保留, 为 0;
pbmpsrc	[IN] 指向存储图片源信息的内存空间, 图片源信息结构体 BMPSRC 如下: <pre> struct tagBitmapSource{ unsigned short type; unsigned short version; unsigned long size; unsigned char* pdata; } JHBMP_SRC; </pre> type: 图片类型, 始终为 0, 表示为文件图片; version: 版本, 当前版本为 0; size: 图片源数据大小; pdata: 图片源数据; data 为动态数组, 大小由 size 决定;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不足

功能描述:

在实例句柄对应控制卡的显示屏指定位置显示一幅静态图片。

注: 如果位图数据大小超过 1200 字节, 可将位图分块进行显示。

80.JHScrollLeft（左移一格）

```
JHZRESULT JHZAPI JHScrollLeft(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
x [IN] 被移动矩形区域左上角起始点 X 坐标。
y [IN] 被移动矩形区域左上角矩形起始点 Y 坐标。
width [IN] 被移动矩形区域矩形宽度。
height [IN] 被移动矩形区域矩形高度。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

将实例句柄对应控制卡的显示屏上由 x, y, width, height 确定的矩形区域的显示内容左移一格。

81.JHScrollRight（右移一格）

```
JHZRESULT JHZAPI JHScrollRight(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
x [IN] 被移动矩形区域左上角矩形起始点 X 坐标。
y [IN] 被移动矩形区域左上角矩形起始点 Y 坐标。
width [IN] 被移动矩形区域矩形宽度。
height [IN] 被移动矩形区域矩形高度。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

将实例句柄对应控制卡的显示屏上由x, y, width, height确定的矩形区域的显示内

容右移一格。

82.JHScrollUp（上移一格）

```
JHZRESULT JHZAPI JHScrollUp(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
x [IN] 被移动矩形区域左上角矩形起始点 X 坐标。
y [IN] 被移动矩形区域左上角矩形起始点 Y 坐标。
width [IN] 被移动矩形区域矩形宽度。
height [IN] 被移动矩形区域矩形高度。

返回值：

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述：

将实例句柄对应控制卡的显示屏上由x, y, width, height确定的矩形区域的显示内容上移一格。

83.JHScrollDown（下移一格）

```
JHZRESULT JHZAPI JHScrollDown(  
    HANDLE hdl,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
x [IN] 被移动矩形区域左上角矩形起始点 X 坐标。
y [IN] 被移动矩形区域左上角矩形起始点 Y 坐标。
width [IN] 被移动矩形区域矩形宽度。
height [IN] 被移动矩形区域矩形高度。

返回值：

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述:

将实例句柄对应控制卡的显示屏上由x, y, width, height确定的矩形区域的显示内容下移一格。

84.JHGetWndCnt（查询分区数目）

```
JHZRESULT JHZAPI JHGetWndCnt(  
    HANDLE hdl,  
    unsigned short * pcnt);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
pcnt [OUT] 指向存储分区数目的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡中已创建的分区数目。

85.JHGetWndId（查询分区编号）

```
JHZRESULT JHZAPI JHGetWndId(  
    HANDLE hdl,  
    unsigned short index,  
    unsigned short * wid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
index [IN] 分区的索引, 有效值为 0 到分区数目-1。
wid [OUT] 指向存储分区编号的内存空间, 分区编号有效值为 1 至最大支持分区数。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

查询实例句柄对应控制卡中索引index锁对应的分区编号。

86.JHFingerWnd（查询分区属性）

```
JHZRESULT JHZAPI JHFingerWnd(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short * px,  
    unsigned short * py,  
    unsigned short * pwidth,  
    unsigned short * pheight,  
    unsigned long * pstyle,  
    unsigned long * puserdata);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为 1 至最大支持分区数。
px	[OUT]指向存储分区起始点 X 坐标的内存空间。
py	[OUT]指向存储分区起始点 Y 坐标的内存空间。
pwidth	[OUT]指向存储分区宽度的内存空间。
pheight	[OUT]指向存储分区高度的内存空间。
pstyle	[OUT]指向存储节目样式的内存空间，格式如下： 位 31-4：保留，始终为 0； 位 3：是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2：加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1：节目属性，为 0 表示临时节目，为 1 表示永久节目； 位 0：保留，始终为 0；
puserdata	[OUT]指向储存用户数据的内存空间。用户数据保留，始终为 0。

注：传入的指针不能为空。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡中指定分区编号的分区属性。

87.JHCreateWnd（创建分区）

```
JHZRESULT JHZAPI JHCreateWnd(  
    HANDLE hdl,  
    unsigned short wid,
```

```
unsigned short option,  
unsigned short x,  
unsigned short y,  
unsigned short width,  
unsigned short height,  
unsigned long style,  
unsigned long userdata);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为 1 至最大支持分区数。
option	[IN] 创建方式，0x0000 表示创建时若分区编号已存在，则创建失败；为 0x0001 时，表示不管是否已存在，均创建(若已存在，则覆盖，覆盖时，会先清除分区原有节目)。
x	[IN] 分区起始点 X 坐标。
y	[IN] 分区起始点 Y 坐标。
width	[IN] 分区宽度。
height	[IN] 分区高度。
style	[IN] 分区样式，格式如下： 位 31- 3: 保留，始终为 0； 位 1: 永久；0 为临时分区，1 为永久分区； 位 0: 使能；保留，始终为 0；
userdata	[IN] 用户数据，保留，始终为 0。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应控制卡上根据 option 确定的创建方式指定分区属性创建编号为 wid 的分区。

88.JHChangeWnd（修改分区属性）

```
JHZRESULT JHZAPI JHChangeWnd(  
HANDLE hdl,  
unsigned short wid,  
unsigned short x,  
unsigned short y,  
unsigned short width,  
unsigned short height,  
unsigned long style,  
unsigned long userdata);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区。

x	[IN] 分区起始点 X 坐标。
y	[IN] 分区起始点 Y 坐标。
width	[IN] 分区宽度。
height	[IN] 分区高度。
style	[IN] 分区样式，格式如下： 位 31- 3：保留 ，始终为 0； 位 1：永久；0 为临时分区，1 为永久分区； 位 0：使能；保留，始终为 0；
userdata	[IN] 用户数据，保留，始终为 0。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡中根据指定的分区属性修改编号为 wid 的分区。

89.JHDeleteWnd（删除分区）

```
JHZRESULT JHZAPI JHDeleteWnd(
    HANDLE hdl,
    unsigned short wid);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

删除实例句柄对应控制卡中分区编号为 wid 的分区。

90.JHGetProgress（查询分区播放状态）

```
JHZRESULT JHZAPI JHGetProgress(
    HANDLE hdl,
    unsigned short wid,
    unsigned long * cur,
    unsigned long * ttl);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区。

cur [OUT]指向存储指定分区已运行时间的内存空间，单位：毫秒。
ttl [OUT]指向存储指定分区总运行所需时间的内存空间，单位：毫秒。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡中根据分区的编号查询该分区的总运行时间和当前已经运行的时间。

91.JHCreateTextProg（创建文本节目）

```
JHZRESULT JHZAPI JHCreateTextProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    const JHTEXTPROG * ptextprog,  
    const char *pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63； 位 7-0: 保留，始终为 0。
style	[IN] 节目样式。 位 31-4: 保留，始终为 0； 位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）； 位 0: 保留，始终为 0；
format	[IN] 文本显示格式，如下所示： 位 31-20: 保留，始终为 0； 位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色） 位 18: 蓝色； 位 17: 绿色；

位 16: 红色;
 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID;
 位 7- 6: 水平对齐 HALIGN
 00: 左对齐;
 01: 水平居中对齐;
 10: 右对齐;
 位 5- 4: 垂直对齐方式;
 00: 上对齐;
 01: 垂直居中对齐;
 10: 下对齐;
 位 3: 保留, 始终为 0;
 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本;
 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0;

ptextprog

[IN] 指向存储 JHTEXTPROG 结构数据的内存空间, 结构成员如下:

tpEntryMode: 进入效果; 0x0000 不合法, 强制转为 0x0001.

0x0001: 立即显示
 0x0002: 向左移入
 0x0003: 向右移入
 0x0004: 向上移入
 0x0005: 向下移入
 0x0006: 向右展开
 0x0007: 向左展开
 0x0008: 向上展开
 0x0009: 向下展开
 0x000A: 左右合并
 0x000B: 左右展开
 0x000C: 上下合并
 0x000D: 画卷右展开
 0x000E: 画卷左展开
 0x000F: 画卷左右展开
 0x0010: 画卷左右合并
 0x0011: 画卷上下合并
 0x0012: 拉开窗帘
 0x0013: 中间溢出
 0x0014: 左右移动
 0x0015: 右下抽出
 0x0016: 右上抽出
 0x0017: 左下抽出
 0x0018: 左上抽出
 0x0019: 水平百叶窗
 0x001A: 垂直百叶窗

0x001B:矩形扩散
0x001C:矩形收缩
0x001D:菱形扩散
0x001E:菱形收缩
0x001F:十字扩散
0x0020:顺时针 1 根轮辐
0x0021:顺时针 2 根轮辐
0x0022:顺时针 4 根轮辐
0x0023:顶端画扇形
0x0024:底端画扇形
0x0025:中间画扇形
0x0026:四顶点画扇形
0x0027:水平梳理
0x0028:垂直梳理
0x0029:左右穿插
0x002A:飘雪
0x002B:冒泡
0x002C:左辐射
0x002D:右辐射
0x002E:上下生长
0x002F:左开始波形
0x0030:右开始波形
0xFFFF:随机进入

tpEntrySpeed: 进入效果速度; 该值越大, 运动速度越慢;

tpEntryStaytime: 进入效果停留时间; 单位 10 毫秒;

tpHighLightMode: 0x0000 时无强调效果, 0x0001 时闪烁;

tpHighLightSpeed: 有强调效果时生效, 闪烁的速度。

tpHighLightStaytime: 强调效果停留时间; 单位 10 毫秒。

tpExitMode: 退出效果。

0x0000:无退出效果

0x0001:立即清屏

0x0002:向左移出

0x0003:向右移出

0x0004:向上移出

0x0005:向下移出

0x0006:左右合并

0x0007:左右展开

0x0008:上下合并

0xFFFF:随机退出

tpExitSpeed: 退出效果速度; 该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。

pformatstring [IN] 指向存储待显示的文本字符串的内存空间。文本字符编码, 可包含有效的 ASCII 码(不能包含 0x00)和 GB2312 编码, 可混用。换行

符为' \n' ,即 0x0A; 水平制表符' \t' ,即 0x09,显示为 4 个 ASCII 空格符。

(注: FORMAT 属性说明, 主要注意两类, 单行文本和多行文本

1. 单行文本时:

忽略 wordbreak 自动换行设置, 无论是自动换行或是手动在命令中添加换行符' \n' , 均忽略, 不作换行处理,

2. 多行文本时:

根据设置不作限制, 自动换行和手动换行可同时支持;

但是若文本显示所需区域超过当前限制区域, 建议做如下处理:

1. 若为单行文本, 建议不要设置水平居中; 其他可任意搭配;

2. 若为多行文本, 若使用了自动换行, 建议不要使用垂直居中对齐; 若无自动换行, 建议不要使用任何居中对齐方式;

使用时若遇到上面情况, 可参照上面说明使用, 否则为了整体显示效果, 系统会自行裁剪部分数据;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应控制卡中根据指定的参数在分区编号为wid的分区上创建文本节目。

92. JHCreateBitmapProg (创建图片节目)

```
JHZRESULT JHZAPI JHCreateBitmapProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    const JHTEXTPROG * ptextprog,  
    unsigned short rev2,  
    const JHBMP SRC * pbmpsrc);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
wid	[IN] 分区编号, 有效值为控制卡中已经创建的分区, 或 0xFFFF; 为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识, 用于单分区多节目。 位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63;

位 7-0: 保留, 始终为 0。

style [IN] 节目样式。

位 31-4: 保留, 始终为 0;

位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载;

位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除;

位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失);

位 0: 保留, 始终为 0;

format [IN] 图片显示格式;

位 31-8: 保留, 为 0;

位 7- 6: 水平对齐 HALIGN

00: 左对齐;

01: 水平居中对齐;

10: 右对齐;

位 5- 4:

00: 上对齐;

01: 垂直居中对齐;

10: 下对齐;

位 3-0: 保留, 为 0;

ptextprog [IN] 指向 JHTEXTPROG 类型的内存空间。JHTEXTPROG 结构体格式如下所示:

tpEntryMode: 进入效果。

0x0001: 立即显示。

tpEntrySpeed: 进入效果速度。该值越大, 运动速度越慢。

tpEntryStaytime: 进入效果停留时间, 单位 10 毫秒。

tpHighLightMode: 强调。

0x0000: 无强调效果

0x0001: 闪烁

tpHighLightSpeed: 强调效果速度。该值越大, 运动速度越慢。

tpHighLightStaytime: 强调效果停留时间, 单位 10 毫秒。

tpExitMode: 退出效果。

0x0000: 无退出效果

0x0001: 立即清屏

0x0002: 向左移出

0x0003: 向右移出

0x0004: 向上移出

0x0005: 向下移出

0x0006: 左右合并

0x0007: 左右展开

0x0008: 上下合并

0xFFFF: 随机退出

tpExitSpeed: 退出效果速度。该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除

非有新的节目过来，否则一直循环播放。

rev2 [IN] 保留；始终为 0；

JHBMP SRC [IN] 图片源信息；

```

struct tagBitmapSource{
    unsigned short type;
    unsigned short version;
    unsigned long size;
    unsigned char *pdata;
} JHBMP SRC;

```

type: 图片类型，始终为 0, 表示为文件图片；

version: 版本，当前版本为 0；

size: 图片源数据大小；

pdata: 图片源数据；data 为动态数组，大小由 size 决定；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在实例句柄对应控制卡中根据指定的参数在分区编号为 wid 的分区上创建图片节目。

注：如果位图数据大小超过 1200 字节，可建立多个分区显示图片。

93.JHCreateQRCodeProg（创建二维码节目）

```

JHZRESULT JHZAPI JHCreateQRCodeProg(
    HANDLE hdl,
    unsigned short wid,
    unsigned short rev,
    unsigned long style,
    unsigned long format,
    const JHTEXTPROG * ptextprog,
    const JHQR CODE* pqr code,
    const char *pformatstring);

```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63；

	位 7-0: 保留, 始终为 0。
style	[IN] 节目样式。 位 31-4: 保留, 始终为 0; 位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载; 位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除; 位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失); 位 0: 保留, 始终为 0;
format	[IN] 二维码图片显示格式, 如下所示: 位 31-8: 保留, 为 0; 位 7- 6: 水平对齐 HALIGN 00: 左对齐; 01: 水平居中对齐; 10: 右对齐; 位 5- 4: 00: 上对齐; 01: 垂直居中对齐; 10: 下对齐; 位 3-0: 保留, 为 0;
ptextprog	[IN] 指向存储 JHTEXTPROG 结构数据的内存空间, 结构成员如下: tpEntryMode: 进入效果; 0x0000 不合法, 强制转为 0x0001. 0x0001: 立即显示 0x0002: 向左移入 0x0003: 向右移入 0x0004: 向上移入 0x0005: 向下移入 0x0006: 向右展开 0x0007: 向左展开 0x0008: 向上展开 0x0009: 向下展开 0x000A: 左右合并 0x000B: 左右展开 0x000C: 上下合并 0x000D: 画卷右展开 0x000E: 画卷左展开 0x000F: 画卷左右展开 0x0010: 画卷左右合并 0x0011: 画卷上下合并 0x0012: 拉开窗帘 0x0013: 中间溢出 0x0014: 左右移动 0x0015: 右下抽出 0x0016: 右上抽出

0x0017:左下抽出
0x0018:左上抽出
0x0019:水平百叶窗
0x001A:垂直百叶窗
0x001B:矩形扩散
0x001C:矩形收缩
0x001D:菱形扩散
0x001E:菱形收缩
0x001F:十字扩散
0x0020:顺时针 1 根轮辐
0x0021:顺时针 2 根轮辐
0x0022:顺时针 4 根轮辐
0x0023:顶端画扇形
0x0024:底端画扇形
0x0025:中间画扇形
0x0026:四顶点画扇形
0x0027:水平梳理
0x0028:垂直梳理
0x0029:左右穿插
0x002A:飘雪
0x002B:冒泡
0x002C:左镭射
0x002D:右镭射
0x002E:上下生长
0x002F:左开始波形
0x0030:右开始波形
0xFFFF:随机进入

tpEntrySpeed: 进入效果速度; 该值越大, 运动速度越慢;

tpEntryStaytime: 进入效果停留时间; 单位 10 毫秒;

tpHighLightMode: 保留, 始终为 0;

tpHighLightSpeed: 保留, 始终为 0。

tpHighLightStaytime: 保留, 始终为 0。

tpExitMode: 退出效果。

0x0000:无退出效果

0x0001:立即清屏

0x0002:向左移出

0x0003:向右移出

0x0004:向上移出

0x0005:向下移出

0x0006:左右合并

0x0007:左右展开

0x0008:上下合并

0xFFFF:随机退出

tpExitSpeed: 退出效果速度; 该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。

Pqrcode [IN] 指向存储 JHQRCODE 结构数据的内存空间, 结构成员如下:

```
typedef struct{
    unsigned short rev;
    unsigned short type;
    unsigned short version;
} JHQRCODE;
```

rev :保留, 始终为 0。

type : 类型, 二维码图片节目值为 2 (位图图片节目值为 0)。

version: 保留, 始终为 0。

pformatstring [IN] 指向存储待显示的文本字符串的内存空间 (1-128 字节)。文本字符编码, 可包含有效的 ASCII 码 (不能包含 0x00)。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应控制卡中根据指定的参数在分区编号为wid的分区上创建二维码节目。

94.JHCreateSimpleDigClockProg (创建简易时钟)

```
JHZRESULT JHAPI JHCreateSimpleDigClockProg(
    HANDLE hdl,
    unsigned short wid,
    unsigned short rev,
    unsigned long style,
    unsigned long format,
    unsigned long timeoffset);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
wid	[IN] 分区编号, 有效值为控制卡中已经创建的分区, 或 0xFFFF; 为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识, 用于单分区多节目。 位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63; 位 7-0: 保留, 始终为 0。
style	[IN] 节目样式。 位 31-4: 保留, 始终为 0; 位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示

按队列顺序加载；

位 2：加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除；

位 1：节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）；

位 0：保留，始终为 0；

format [IN] 文本显示格式。

位 31-20：保留，始终为 0。

位 19- 16：颜色（注：0000 表示使用系统当前颜色；1000 为黑色）

位 18：蓝色。

位 17：绿色。

位 16：红色。

位 15- 8：字体 ID；有效值为 0-254；为 0 时使用当前字体 ID。

位 7- 6：水平对齐 HALIGN

00：左对齐。

01：水平居中对齐。

10：右对齐。

位 5- 4：垂直对齐方式。

00：上对齐。

01：垂直居中对齐。

10：下对齐。

位 3：保留，始终为 0。

位 2：wordbreak，自动换行，为 0，不自动换行，为 1，自动换行；

位 1：单行文本； 0 表示多行文本，1 表示单行文本。

位 0：高级文本，0 表示普通文本，1 表示高级文本；保留，始终为 0。

timeoffset [IN] 时间偏移，单位秒，可用于实现多时区时间显示，有符号数值。

注：FORMAT 属性说明，主要注意两类，单行文本和多行文本

1. 单行文本时：

忽略 wordbreak 自动换行设置，无论是自动换行或是手动在命令中添加换行符' \n'，均忽略，不作换行处理，

2. 多行文本时：

根据设置不作限制，自动换行和手动换行可同时支持；

但是若文本显示所需区域超过当前限制区域，建议做如下处理：

1. 若为单行文本，建议不要设置水平居中；其他可任意搭配；

2. 若为多行文本，若使用了自动换行，建议不要使用垂直居中对齐；若无自动换行，建议不要使用任何居中对齐方式；

使用时若遇到上面情况，可参照上面说明使用，否则为了整体显示效果，系统会自行裁剪部分数据；

返回值：

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述：

在实例句柄对应控制卡中根据指定的参数在分区编号为 wid 的分区上创建创建简

易时钟节目。

95.JHCreateDigClockProg（创建数字时钟）

```
JHZRESULT JHZAPI JHCreateDigClockProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset,  
    const char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63； 位 7-0: 保留，始终为 0。
style	[IN] 节目样式。 位 31-4: 保留，始终为 0； 位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）； 位 0: 保留，始终为 0；
format	[IN] 文本显示格式。 位 31-20: 保留，始终为 0。 位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色） 位 18: 蓝色。 位 17: 绿色。 位 16: 红色。 位 15- 8: 字体 ID；有效值为 0-254；为 0 时使用当前字体 ID。 位 7- 6: 水平对齐 HALIGN 00: 左对齐。 01: 水平居中对齐。 10: 右对齐。 位 5- 4: 垂直对齐方式 00: 上对齐。 01: 垂直居中对齐。

	10: 下对齐。
	位 3: 保留, 始终为 0。
	位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
	位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。
	位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。
timeoffset	[IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。
pformatstring	[IN] 指向存储文本格式化字符串的内存空间。文本格式化字符串的格式如下(注意, 下面描述中注意大小写的形式, 若须要显示 “%”, 则在 sFORMAT 中需要填补两个百分号, 即 “%%”; 换行符为 ‘\n’, 即 0x0A; 水平制表符 ‘\t’, 即 0x09, 显示为 4 个 ASCII 空格符): 年: %yyyy 或 %yy, 分别用于显示四位年份或两位年份; 月: %M 或 %MM; %M 为月份是几位就显示几位, %MM 为不足两位数的月份, 前面补零; 日: %d 或 %dd; %d 为日期是几位就显示几位, %dd 为不足两位数的日期, 前面补零; 时: %H, %HH, %h, %hh; H 是 24 小时制, h 是 12 小时制; %H 和 %h 为时钟是几位就显示几位, %HH 和 %hh 为不足两位数的时钟, 前面补零; 分: %m 或 %mm; %m 为分钟是几位就显示几位, %mm 为不足两位数的分钟, 前面补零; 秒: %s 或 %ss; %s 为秒钟是几位就显示几位, %ss 为不足两位数的秒钟, 前面补零; 星期: %w 或 %W; %w 为英文星期缩写; %W 为中文星期数; %w: Sun; Mon; Tue; Wed; Thu; Fri; Sat; %W: 星期日; 星期一; 星期二; 星期三; 星期四; 星期五; 星期六; 上/下午: %t 或 %T; %t 为英文, AM 或 PM; %T 为中文, 上午或下午; 一个文本格式化字符串可以为 “%yyyy 年 %MM 月 %dd 日 %HH 时 %mm 分 %ss 秒 %W”, 年、月、日、时、分、秒、星期都是可选的, 例如, 如果不需要在显示屏上显示年份, 文本格式化字符串改为 “%MM 月 %dd 日 %HH 时 %mm 分 %ss 秒 %W”。

注: FORMAT 属性说明, 主要注意两类, 单行文本和多行文本

1. 单行文本时:

忽略 wordbreak 自动换行设置, 无论是自动换行或是手动在命令中添加换行符 ‘\n’, 均忽略, 不作换行处理,

2. 多行文本时:

根据设置不作限制, 自动换行和手动换行可同时支持;

但是若文本显示所需区域超过当前限制区域, 建议做如下处理:

1. 若为单行文本, 建议不要设置水平居中; 其他可任意搭配;

2. 若为多行文本, 若使用了自动换行, 建议不要使用垂直居中对齐; 若无自动换行, 建议

不要使用任何居中对齐方式；
使用时若遇到上面情况，可参照上面说明使用，否则为了整体显示效果，系统会自行裁剪部分数据；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡中根据指定的参数在分区编号为 wid 的分区上创建高级时钟节目。

96.JHCreateComplexDigClockProg（创建高级数字时钟）

```
JHZRESULT JHZAPI JHCreateComplexDigClockProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset,  
    const JHTEXTPROG * ptextprog,  
    const char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63； 位 7-0: 保留，始终为 0。
style	[IN] 节目样式。 位 31-4: 保留，始终为 0； 位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）； 位 0: 保留，始终为 0；
format	[IN] 文本显示格式。 位 31-20: 保留，始终为 0。 位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色） 位 18: 蓝色。 位 17: 绿色。

	位 16: 红色。
	位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。
	位 7- 6: 水平对齐 HALIGN
	00: 左对齐。
	01: 水平居中对齐。
	10: 右对齐。
	位 5- 4: 垂直对齐方式
	00: 上对齐。
	01: 垂直居中对齐。
	10: 下对齐。
	位 3: 保留, 始终为 0。
	位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
	位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。
	位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。
timeoffset	[IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。
ptextprog	[IN] 指向存储 JHTEXTPROG 结构数据的内存空间, 结构成员如下:
	tpEntryMode: 进入效果; 0x0000 不合法, 强制转为 0x0001.
	0x0001: 立即显示
	0x0002: 向左移入
	0x0003: 向右移入
	0x0004: 向上移入
	0x0005: 向下移入
	0x0006: 向右展开
	0x0007: 向左展开
	0x0008: 向上展开
	0x0009: 向下展开
	0x000A: 左右合并
	0x000B: 左右展开
	0x000C: 上下合并
	0x000D: 画卷右展开
	0x000E: 画卷左展开
	0x000F: 画卷左右展开
	0x0010: 画卷左右合并
	0x0011: 画卷上下合并
	0x0012: 拉开窗帘
	0x0013: 中间溢出
	0x0014: 左右移动
	0x0015: 右下抽出
	0x0016: 右上抽出
	0x0017: 左下抽出
	0x0018: 左上抽出
	0x0019: 水平百叶窗
	0x001A: 垂直百叶窗

0x001B:矩形扩散
 0x001C:矩形收缩
 0x001D:菱形扩散
 0x001E:菱形收缩
 0x001F:十字扩散
 0x0020:顺时针 1 根轮辐
 0x0021:顺时针 2 根轮辐
 0x0022:顺时针 4 根轮辐
 0x0023:顶端画扇形
 0x0024:底端画扇形
 0x0025:中间画扇形
 0x0026:四顶点画扇形
 0x0027:水平梳理
 0x0028:垂直梳理
 0x0029:左右穿插
 0x002A:飘雪
 0x002B:冒泡
 0x002C:左辐射
 0x002D:右辐射
 0x002E:上下生长
 0x002F:左开始波形
 0x0030:右开始波形
 0xFFFF:随机进入

tpEntrySpeed: 进入效果速度; 该值越大, 运动速度越慢;

tpEntryStaytime: 进入效果停留时间; 单位 10 毫秒;

tpHighLightMode: 0x0000 时无强调效果, 0x0001 时闪烁;

tpHighLightSpeed: 有强调效果时生效, 闪烁的速度。

tpHighLightStaytime: 强调效果停留时间; 单位 10 毫秒。

tpExitMode: 退出效果。

0x0000:无退出效果
 0x0001:立即清屏
 0x0002:向左移出
 0x0003:向右移出
 0x0004:向上移出
 0x0005:向下移出
 0x0006:左右合并
 0x0007:左右展开
 0x0008:上下合并
 0xFFFF:随机退出

tpExitSpeed: 退出效果速度; 该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。

pformatstring [IN] 指向存储文本格式化字符串的内存空间。文本格式化字符串的格式如下(注意, 下面描述中注意大小写的形式, 若须要显示 “%”,

则在 sFORMAT 中需要填补两个百分号，即 “%%”；换行符为 ‘\n’，即 0x0A；水平制表符 ‘\t’，即 0x09，显示为 4 个 ASCII 空格符)：

年：%yyyy 或 %yy，分别用于显示四位年份或两位年份；

月：%M 或 %MM；%M 为月份是几位就显示几位，%MM 为不足两位数的月份，前面补零；

日：%d 或 %dd；%d 为日期是几位就显示几位，%dd 为不足两位数的日期，前面补零；

时：%H，%HH，%h，%hh；H 是 24 小时制，h 是 12 小时制；%H 和 %h 为时钟是几位就显示几位，%HH 和 %hh 为不足两位数的时钟，前面补零；

分：%m 或 %mm；%m 为分钟是几位就显示几位，%mm 为不足两位数的分钟，前面补零；

秒：%s 或 %ss；%s 为秒钟是几位就显示几位，%ss 为不足两位数的秒钟，前面补零；

星期：%w 或 %W；%w 为英文星期缩写；%W 为中文星期数；

 %w: Sun; Mon; Tue; Wed; Thu; Fri; Sat;

 %W: 星期日; 星期一; 星期二; 星期三; 星期四; 星期五; 星期六;

 上/下午：%t 或 %T；%t 为英文，AM 或 PM；%T 为中文，上午或下午；

一个文本格式化字符串可以为“%yyyy 年 %MM 月 %dd 日 %HH 时 %mm 分 %ss 秒 %W”，年、月、日、时、分、秒、星期都是可选的，例如，如果不需要在显示屏上显示年份，文本格式化字符串改为“%MM 月 %dd 日 %HH 时 %mm 分 %ss 秒 %W”。

注：FORMAT 属性说明，主要注意两类，单行文本和多行文本

1. 单行文本时：

忽略 wordbreak 自动换行设置，无论是自动换行或是手动在命令中添加换行符 ‘\n’，均忽略，不作换行处理，

2. 多行文本时：

根据设置不作限制，自动换行和手动换行可同时支持；

但是若文本显示所需区域超过当前限制区域，建议做如下处理：

1. 若为单行文本，建议不要设置水平居中；其他可任意搭配；
2. 若为多行文本，若使用了自动换行，建议不要使用垂直居中对齐；若无自动换行，建议不要使用任何居中对齐方式；

使用时若遇到上面情况，可参照上面说明使用，否则为了整体显示效果，系统会自行裁剪部分数据；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡中根据指定的参数在分区编号为 wid 的分区上创建高级数

字时钟节目。

97.JHCreateCountdownProg（倒计时节目）

```
JHZRESULT JHZAPI JHCreateCountdownProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset,  
    unsigned long flag,  
    const JHSYSTEMTIME * ptime,  
    const char * pformatstring);
```

hdl [IN] 实例句柄，与控制卡相对应。

wid [IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。

rev [IN] 节目标识，用于单分区多节目。
位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63；
位 7-0: 保留，始终为 0。

style [IN] 节目样式。
位 31-4: 保留，始终为 0；
位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载；
位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除；
位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）；
位 0: 保留，始终为 0；

format [IN] 文本显示格式。
位 31-20: 保留，始终为 0。
位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色）
位 18: 蓝色。
位 17: 绿色。
位 16: 红色。
位 15- 8: 字体 ID；有效值为 0-254；为 0 时使用当前字体 ID。
位 7- 6: 水平对齐 HALIGN
00: 左对齐。
01: 水平居中对齐。
10: 右对齐。
位 5- 4: 垂直对齐方式

00: 上对齐。
01: 垂直居中对齐。
10: 下对齐。

位 3: 保留, 始终为 0。

位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;

位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。

位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。

timeoffset [IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。

flag [IN] 控制信息

位 31-1: 保留, 必须为 0;

位 0: 参考时间类型, 为 0 表示 ptime 为绝对时间, 为 1 表示 ptime 为相对当前的时间;

注: 当 FLAG 的位 0 为 1 时, ptime 中只有 wDay、wHour、wMinute、wSecond 有效, 参考的时间点为加载时的时间和这四个值所表示的时间之和 ((wDay*86400) + (wHour*3600) + (wMinute*60) + wSecond) 。

ptime [IN] 指向 JHSYSTEMTIME 类型的内存空间。JHSYSTEMTIME 结构的格式如下所示:

wYear: 年

wMonth: 月, 有效值为 1-12;

wDayOfWeek: 星期, 保留值, 始终为 0

wDay: 日, 有效值为 1-31;

wHour: 时, 有效值为 0-23;

wMinute: 分, 有效值为 0-59;

wSecond: 秒, 有效值为 0-59;

wMilliseconds: 毫秒, 保留值, 始终为 0;

注: 表示截止时间 (倒计时到何时结束), 其类型由 FLAG 的位 0 决定。

pformatstring [IN] 指向存储待显示的时间格式字符串内存空间。时间格式化字符串格式如下:

不能包含 0x00; (注意, 下面描述中注意大小写的形式)

日: %D, 总的天数;

时: %H 或 %h, %H 为总的时钟数, 有几位就显示几位; %h 为当前天中的时钟数, 不足两位数的时钟, 前面补零;

分: %M 或 %m; %M 为总的分钟数, 有几位就显示几位; %m 为当前小时中的分钟数, 不足两位数的分钟, 前面补零;

秒: %S 或 %s; %S 为总的秒钟数, 有几位就显示几位; %s 为当前分钟中的秒数, 不足两位数的秒钟, 前面补零;

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能描述:

在实例句柄对应控制卡中根据指定的参数在分区编号为 wid 的分区上创建倒计时节目。

98. JHCreateCountdownProg（正计时节目）

```
JHZRESULT JHZAPI JHCreateCountupProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset,  
    unsigned long flag,  
    const JHSYSTEMTIME * ptime,  
    const char * pformatstring);
```

hdl [IN] 实例句柄，与控制卡相对应。

wid [IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。

rev [IN] 节目标识，用于单分区多节目。
位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63；
位 7-0: 保留，始终为 0。

style [IN] 节目样式。
位 31-4: 保留，始终为 0；
位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载；
位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除；
位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）；
位 0: 保留，始终为 0；

format [IN] 文本显示格式。
位 31-20: 保留，始终为 0。
位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色）
位 18: 蓝色。
位 17: 绿色。
位 16: 红色。
位 15- 8: 字体 ID；有效值为 0-254；为 0 时使用当前字体 ID。
位 7- 6: 水平对齐 HALIGN
00: 左对齐。
01: 水平居中对齐。
10: 右对齐。
位 5- 4: 垂直对齐方式
00: 上对齐。
01: 垂直居中对齐。
10: 下对齐。

位 3: 保留, 始终为 0。
 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。
 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。

timeoffset [IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。
 flag [IN] 控制信息
 位 31-1: 保留, 必须为 0;
 位 0: 参考时间类型, 为 0 表示 ptime 为绝对时间, 为 1 表示 ptime 为相对当前的时间;

注: 当 FLAG 的位 0 为 1 时, ptime 中只有 wDay、wHour、wMinute、wSecond 有效, 参考的时间点为加载时的时间和这四个值所表示的时间之和 ((wDay*86400) + (wHour*3600) + (wMinute*60) + wSecond)。

ptime [IN] 指向 JHSYSTEMTIME 类型的内存空间。JHSYSTEMTIME 结构的格式如下所示:
 wYear: 年
 wMonth: 月, 有效值为 1-12;
 wDayOfWeek: 星期, 保留值, 始终为 0
 wDay: 日, 有效值为 1-31;
 wHour: 时, 有效值为 0-23;
 wMinute: 分, 有效值为 0-59;
 wSecond: 秒, 有效值为 0-59;
 wMilliseconds: 毫秒, 保留值, 始终为 0;

注: ptime 表示起始时间 (从何时开始计时), 其类型由 FLAG 的位 0 决定; 若将其设置为 NULL 则表示 ptime 的所有值为 0。

pformatstring [IN] 指向存储待显示的时间格式字符串内存空间。时间格式化字符串格式如下:
 不能包含 0x00; (注意, 下面描述中注意大小写的形式)
 日: %D, 总的天数;
 时: %H 或 %h, %H 为总的时钟数, 有几位就显示几位; %h 为当前天中的时钟数, 不足两位数的时钟, 前面补零;
 分: %M 或 %m; %M 为总的分钟数, 有几位就显示几位; %m 为当前小时中的分钟数, 不足两位数的分钟, 前面补零;
 秒: %S 或 %s; %S 为总的秒钟数, 有几位就显示几位; %s 为当前分钟中的秒数, 不足两位数的秒钟, 前面补零;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应控制卡中根据指定的参数在分区编号为 wid 的分区上创建正计时节目。

注: 若将 flag 设置为 1 即相对当前时间, 则可用于比如临时计时、计算运行时间等用途。

99.JHFingerProg（获取默认节目）

```
JHZRESULT JHZAPI JHFingerProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short option,  
    unsigned short * prev,  
    unsigned short * ptype,  
    unsigned long * pstyle,  
    char * dat,  
    unsigned short * psize);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区。
option	[IN] 操作方式，格式如下： 位 15-8：节目编号，有效范围 0~63； 位 7-3：保留； 位 2：节目选择，0 表示按位 0 选择，1 表示永久节目和临时节目； 位 1：优先查询选择，0 表示不启用，1 表示启用（位 0 为 0 时，优选查询临时节目，若临时节目不存在，则查询永久节目；位 0 为 1 时，优选查询永久节目，若永久节目不存在，则查询临时节目）； 位 0：是否永久节目，0 表示临时节目，1 表示永久节目；
prev	[OUT] 节目标识，用于单分区多节目。 位 15-8：节目编号，有效范围 0~63； 位 7-0：保留，始终为 0；
ptype	[OUT] 指向存储节目类型的内存空间，节目类型包括： 0x0001—文本节目 0x0002—位图节目 0x0003—保留 0x0004—简易数字时钟 0x0005—数字时钟 0x0006—保留 0x0007—倒计时节目 0x0008—正计时节目 0x0009—表格节目
pstyle	[OUT] 指向存储节目样式的内存空间，格式如下： 位 31-4：保留，始终为 0； 位 3：是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2：加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1：节目属性，为 0 表示临时节目，为 1 表示永久节目； 位 0：保留，始终为 0；
dat	[OUT] 指向存储节目内容的内存空间，长度由 psize 指定，该段由 ptype 的值

（节目类型）决定，如下所示：

0x0001—文本节目对应 CONTENT：

FORMAT	ENTRY	SPENTRY	DUENTRY	HIGHLIGHT	SPHL	DUHL	EXIT
4 字节	2 字节	2 字节	2 字节	2 字节	2 字节	2 字节	2 字节

SPEXIT	TIMES	CNT	TEXT
2 字节	2 字节	2 字节	1~1200 字节

0x0002—位图节目对应 CONTENT：

FORMAT	ENTRY	SPENTRY	DUENTRY	HIGHLIGHT	SPHL	DUHL	EXIT
4 字节	2 字节	2 字节	2 字节	2 字节	2 字节	2 字节	2 字节

SPEXIT	TIMES	POSITION	BITCOUNT	COLORUSED	OFFBITS	BMPWIDTH
2 字节	2 字节	1 字节	1 字节	2 字节	2 字节	2 字节

BMPHEIGHT	BMPCONTENT
2 字节	0~1200 字节

0x0004—简易时钟对应 CONTENT：

FORMAT	OFFSET
4 字节	2 字节

0x0005—数字时钟对应 CONTENT：

FORMAT	OFFSET	CNT	sFORMAT
4 字节	2 字节	2 字节	0~256 字节

0x0007—倒计时节目对应 CONTENT：

FORMAT	OFFSET	FLAG	TIME	CNT	sFORMAT
4 字节	4 字节	4 字节	16 字节	2 字节	2~256 字节

0x0008—正计时节目对应 CONTENT：

FORMAT	OFFSET	FLAG	TIME	CNT	sFORMAT
4 字节	4 字节	4 字节	16 字节	2 字节	2~256 字节

0x0009—表格节目对应 CONTENT：

FORMAT	FLAG	CNT	sFORMAT
4 字节	4 字节	2 字节	1~1200 字节

psize [OUT] 指向存储的节目内容长度的内存空间。

注：参数的具体信息请参考《ZkLED 字库卡 v4. x 使用手册》中“获取分区节目”命令。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获取实例句柄对应控制卡中分区编号为 wid 的分区的节目。

100. JHDeleteProg（删除分区节目）

```
JHZRESULT JHZAPI JHDeleteProg(  
    HANDLE hdl,  
    unsigned short wid,  
    unsigned short option);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

wid [IN] 分区编号，有效值为控制卡中已经创建的分区，若为 0xFFFF 时，表示删除所有分区的节目。

option [IN] 操作选项，**0xFFFF 表示删除分区所有节目。**
位 15-8：节目编号，有效范围 0~63 和 255，其中 255 表示所有节目；
位 7-3：保留；
位 2：节目选择，0 表示按位 0 选择
位 1：保留，始终为 0；
位 0：是否永久节目，0 表示临时节目，1 表示永久节目；

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

删除实例句柄对应控制卡中分区编号为 wid 的分区节目。

101. JHGetBatchCnt（查询批处理文件数目）

```
JHZRESULT JHZAPI JHGetBatchCnt(  
    HANDLE hdl,  
    unsigned char * pcnt);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

pcnt [OUT] 指向存储批处理文件数目的内存空间。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

获得实例句柄对应控制卡中批处理文件的数目。

102. JHGetBatchName（查询批处理文件名）

```
JHZRESULT JHZAPI JHGetBatchName(  
    HANDLE hdl,  
    unsigned char index,  
    char * pformatstring);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
index	[IN] 批处理文件索引，有效值从 0x00 开始，小于控制卡系统中当前批处理数目。
pformatstring	[OUT] 指向存储批处理文件名格式化字符串的内存空间，内存空间长度为 48 字节。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应控制卡中查询索引为 index 的批处理文件的文件名。

103. JHPreGetBatch（查询批处理文件内容第一步获取批处理文件总大小）

```
JHZRESULT JHZAPI JHPreGetBatch(  
    HANDLE hdl,  
    HANDLE * fd,  
    const char * pformatstring,  
    unsigned long * readsize);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
fd	[IN] 指向查询批处理文件内容第一步申请的存储查询过程信息的内存空间，查询操作第二步需要根据此信息进行操作，查询操作第三步释放该内存空间。
pformatstring	[IN] 指向存储批处理文件名格式化字符串的内存空间。
readsize	[OUT] 指向存储批处理文件总大小的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不够

功能描述:

查询批处理文件内容第一步, 申请一块内存保存查询操作过程的信息, 提供给查询操作第二步使用。

104. JHReadBatch (查询批处理文件内容第二步获取批处理文件数据)

```
JHZRESULT JHZAPI JHReadBatch(  
    HANDLE hdl,  
    HANDLE fd,  
    unsigned long roffset,  
    unsigned char * pdat,  
    unsigned long *psize);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
fd [IN] 指向查询批处理文件内容第一步申请的存储查询过程信息的内存空间, 查询操作第二步需要根据此信息进行操作, 查询操作第三步释放该内存空间。
roffset [IN] 查询的数据在批处理文件中的偏移。
pdat [OUT] 指向存储查询到的数据的内存空间。
psize [OUT] 指向存储查询到的批处理文件数据长度的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在实例句柄对应的控制卡中查询JHPreGetBatch的pformatstring指定的批处理文件的内容, 批处理文件较大时, 需分包发送。

105. JHFinGetBatch (查询批处理文件内容第三步结束查询, 释放资源)

```
JHZRESULT JHZAPI JHFinGetBatch(  
    HANDLE hdl,  
    HANDLE fd);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

fd [IN] 指向查询批处理文件内容第一步申请的存储查询过程信息的内存空间，查询操作第二步需要根据此信息进行操作，查询操作第三步释放该内存空间。

返回值:

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述:

批处理文件内容查询结束，释放fd内存空间

106. JHInquireBatch（查询批处理文件是否存在）

```
JHZRESULT JHZAPI JHInquireBatch(  
    HANDLE hdl,  
    const char * pformatstring);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
pformatstring [IN] 指向存储批处理文件名格式化字符串的内存空间。

返回值:

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述:

查询实例句柄对应控制卡中pformatstring确定的文件是否存在。

107. JHPreCreateBatch（添加批处理文件内容第一步发送 批处理文件总大小）

```
JHZRESULT JHZAPI JHPreCreateBatch(  
    HANDLE hdl,  
    HANDLE * fd,  
    const char * pformatstring,  
    unsigned long writesize);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
fd [IN] 指向添加批处理文件内容第一步申请的存储查询过程信息的内存空间，添加操作第二步需要根据此信息进行操作，添加操作第三步释放该内存空间。
pformatstring [IN] 指向存储批处理文件名格式化字符串的内存空间。
writesize [IN] 批处理文件总长度。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不够

功能描述：

添加批处理文件到实例句柄对应控制卡中的第一步操作。

108. JHWriteBatch（添加批处理文件内容第二步发送批处理文件数据）

```
JHZRESULT JHZAPI JHWriteBatch(
    HANDLE hdl,
    HANDLE fd,
    unsigned long woffset,
    unsigned char * pdat,
    unsigned long size);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

fd [IN] 指向添加批处理文件内容第一步申请的存储查询过程信息的内存空间，添加操作第二步需要根据此信息进行操作，添加操作第三步释放该内存空间。

woffset [IN] 所要发送批处理文件数据在批处理文件中的偏移。

pdat [IN] 指向存储本次所要发送批处理文件数据的内存空间。

size [IN] 指向存储本次所要发送批处理文件数据长度的内存空间。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

添加批处理文件到实例句柄对应控制卡中的第二步操作。

109. JHFinCreateBatch（添加批处理文件内容第三步结束添加，释放资源）

```
JHZRESULT JHZAPI JHFinCreateBatch(
    HANDLE hdl,
    HANDLE fd);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

fd [IN] 指向添加批处理文件内容第一步申请的存储查询过程信息的内存空间, 添加操作第二步需要根据此信息进行操作, 添加操作第三步释放该内存空间。

返回值:

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述:

添加批处理文件到实例句柄对应控制卡中的第三步操作, 释放 fd 资源。

110. JHDeleteBatch (删除批处理)

```
JHZRESULT JHZAPI JHDeleteBatch(  
    HANDLE hdl,  
    const char * pformatstring);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

pformatstring [IN] 指向存储批处理文件名格式化字符串的内存空间。如果包含多个批处理文件名, 批处理文件名间以逗号 “|” 隔开, 最后一个文件名以 0x00 结束; 每个批处理文件名大小范围均为 2~48 字节; **需至少包含一个批处理文件, 否则配置失败。**

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能描述:

删除实例句柄对应控制卡中 formatstring 指定的批处理文件。

111. JHModifyBatchName (修改批处理文件名)

```
JHZRESULT JHZAPI JHModifyBatchName(  
    HANDLE hdl,  
    const char * poformatstring,  
    char * pnformatstring);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

poformatstring [IN] 指向存储原批处理文件名格式化字符串的内存空间。

pnformatstring [OUT] 指向存储新批处理文件名格式化字符串的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

修改实例句柄对应控制卡中的批处理文件名。

112. JHExecuteBatch(调用批处理文件)

```
JHZRESULT JHZAPI JHExecuteBatch(
    HANDLE hdl,
    const char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pformatstring	[IN] 指向存储批处理文件名格式化字符串的内存空间。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节； 需至少包含一个批处理文件，否则配置失败。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

调用实例句柄对应控制卡中的批处理（执行批处理文件里的内容）。

113. JHEventPoweronConfig（开机事件设置）

```
JHZRESULT JHZAPI JHEventPoweronConfig(
    HANDLE hdl,
    const char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pformatstring	[IN] 指向存储批处理文件名格式化字符串的内存空间。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节； 需至少包含一个批处理文件，否则配置失败。

注：配置成功后，系统会立即开启事件。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误

JR_INVALID_OBJECT 无效的对象
功能描述：
 用于指定开机时调用的批处理文件。

114. JHEventPoweronDetail（开机事件查询）

```
JHZRESULT JHZAPI JHEventPoweronDetail(  
    HANDLE hdl,  
    unsigned char * on,  
    char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
on	[OUT] 指向存储开关状态的内存空间，开关状态的有效值 0x00 表示关闭，为 0x01 表示开启。
pformatstring	[OUT] 指向存储批处理文件名格式化字符串的内存空间。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节； 需至少包含一个批处理文件，否则配置失败。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

 查询开机事件关联的批处理文件。

115. JHEventPoweronSwitch（开机事件开关）

```
JHZRESULT JHZAPI JHEventPoweronSwitch(  
    HANDLE hdl,  
    unsigned char on);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
on	[IN] 开关状态，有效值 0x00 表示关闭，为 0x01 表示开启。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

 控制开机事件的开启或关闭状态。

116. JHEventPoweronDelete（开机事件删除）

```
JHZRESULT JHZAPI JHEventPoweronDelete(  
    HANDLE hdl);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

返回值：

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述：

删除开机事件。

117. JHEventNocommConfig（无通讯事件设置）

```
JHZRESULT JHZAPI JHEventNocommConfig(  
    HANDLE hdl,  
    unsigned long time,  
    const char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
time	[IN] 无通讯时间，单位为秒。
pformatstring	[IN] 指向批处理文件名格式化字符串的指针。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节；需至少包含一个批处理文件，否则配置失败。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

设置无通讯事件。超过 time 指定的时间后仍然没有与上位机通讯，则会执行 pformatstring 指定的批处理文件。

118. JHEventNocommDetail（无通讯事件查询）

```
JHZRESULT JHZAPI JHEventNocommDetail(  
    HANDLE hdl,  
    unsigned char * on,
```

```
unsigned long * time,  
char * pformatstring);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
on	[OUT]指向存储开关状态的内存空间。开关状态的有效值 0x00 表示关闭，为 0x01 表示开启。
time	[OUT]指向存储无通讯时间的内存空间。时间单位：秒。
pformatstring	[OUT]指向批处理文件名格式化字符串的指针。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节；需至少包含一个批处理文件，否则配置失败。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

查询无通讯事件。

119. JHEventNocommSwitch（无通讯事件开关）

```
JHZRESULT JHZAPI JHEventNocommSwitch(  
HANDLE hdl,  
unsigned char on);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
on	[IN] 开关状态，有效值 0x00 表示关闭，为 0x01 表示开启。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

控制无通讯事件的开启或关闭状态。

120. JHEventNocommDelete（无通讯事件删除）

```
JHZRESULT JHZAPI JHEventNocommDelete(  
HANDLE hdl);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
-----	--------------------

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

删除无通讯事件关联的批处理文件。

121. JHEventTimerConfig（定时事件设置）

```
JHZRESULT JHZAPI JHEventTimerConfig(  
    HANDLE hdl,  
    unsigned short index,  
    unsigned char week,  
    unsigned char hour,  
    unsigned char minute,  
    const char * pformatstring);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
index	[IN] 定时事件编号，有效值为 1 和 2。
week	[IN] 定时星期，若为 0x00，则为一次性定时事件，事件执行过一次后不会再执行；若为其他值，则为周期性定时事件； 位 0：周一； 位 1：周二； 位 2：周三； 位 3：周四； 位 4：周五； 位 5：周六； 位 6：周日； 位 7：保留位，始终为 0。
hour	[IN] 定时时钟，有效值为 0x00-0x17(即 0 时到 23 时)。
minute	[IN] 定时分钟，有效值为 0x00-0x3B(即 0 分到 59 分)。
pformatstring	[IN] 指向存储批处理文件名格式化字符串的内存空间。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节； 需至少包含一个批处理文件，否则配置失败。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

配置定时事件关联的批处理文件。

122. JHEventTimerDetail（定时事件查询）

```
JHZRESULT JHZAPI JHEventTimerDetail(  
    HANDLE hdl,  
    unsigned short index,  
    unsigned char * on,  
    unsigned char * week,  
    unsigned char * hour,  
    unsigned char * minute,  
    char * pformatstring);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
index	[IN] 定时事件编号，有效值为 1 和 2。
on	[OUT]指向存储开关状态的内存空间。开关状态的有效值 0x00 表示关闭，为 0x01 表示开启。
week	[OUT]指向存储定时星期的内存空间。定时星期的有效值若为 0x00，则为一次性定时事件，事件执行过一次后不会再执行；若为其他值，则为周期性定时事件： 位 0：周一； 位 1：周二； 位 2：周三； 位 3：周四； 位 4：周五； 位 5：周六； 位 6：周日； 位 7：保留位，始终为 0
hour	[OUT] 指向存储定时时钟的内存空间。定时时钟的有效值为 0x00-0x17(即 0 时到 23 时)。
minute	[OUT] 指向存储定时分钟的内存空间。定时分钟的有效值为 0x00-0x3B(即 0 分到 59 分)。
pformatstring	[OUT]指向存储批处理文件名格式化字符串的内存空间。如果包含多个批处理文件名，批处理文件名间以逗号“ ”隔开，最后一个文件名以 0x00 结束；每个批处理文件名大小范围均为 2~48 字节；需至少包含一个批处理文件，否则配置失败。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

查询定时事件关联的批处理文件。

123. JHEventTimerSwitch（定时事件开关）

```
JHZRESULT JHZAPI JHEventTimerSwitch(  
    HANDLE hdl,  
    unsigned short index,  
    unsigned char on);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 定时事件编号，有效值为 1 和 2。
on [IN] 开关状态，有效值 0x00 表示关闭，为 0x01 表示开启。

返回值:

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述:

控制定时事件的开启或关闭状态。

124. JHEventTimerDelete（定时事件删除）

```
JHZRESULT JHZAPI JHEventTimerDelete(  
    HANDLE hdl,  
    unsigned short index);
```

参数:

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 定时事件编号，有效值为 1 和 2。

返回值:

JR_OK 成功
JR_INVALID_OBJECT 无效的对象

功能描述:

删除定时事件关联的批处理文件。

125. JHEventAutoPowerConfig（自动开关机配置）

```
JHZRESULT JHZAPI JHEventAutoPowerConfig(  
    HANDLE hdl,  
    unsigned char on,  
    unsigned char week,  
    unsigned char on_hour,  
    unsigned char on_minute,
```

```

        unsigned char off_hour,
        unsigned char off_minute);

```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
on	[IN] 开机状态, 有效值 0x00 表示关闭, 为 0x01 表示开启。
week	[IN] 星期, 若为 0x00, 表示每天都有该功能; 若为其他, 根据设置开关: 位 0: 周一; 若为 1, 表示每周一都有该功能; 位 1: 周二; 若为 1, 表示每周二都有该功能; 位 2: 周三; 若为 1, 表示每周三都有该功能; 位 3: 周四; 若为 1, 表示每周四都有该功能; 位 4: 周五; 若为 1, 表示每周五都有该功能; 位 5: 周六; 若为 1, 表示每周六都有该功能; 位 6: 周日; 若为 1, 表示每周日都有该功能; 位 7: 保留, 始终为 0;
on_hour	[IN] 开机时钟。
on_minute	[IN] 开机分钟。
off_hour	[IN] 关机时钟。
off_minute	[IN] 关机分钟。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

配置自动开关机。

126. JHEventAutoPowerDetail (自动开关机查询)

```

JHZRESULT JHZAPI JHEventAutoPowerDetail(
    HANDLE hdl,
    unsigned char * on,
    unsigned char * week,
    unsigned char * on_hour,
    unsigned char * on_minute,
    unsigned char * off_hour,
    unsigned char * off_minute);

```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
on	[IN] 指向存储开机状态的内存空间。有效值为 0x00 表示自动开关机处于关闭状态, 为 0x01 表示处于打开状态
week	[IN] 指向存储星期的内存空间。有效值若为 0x00, 表示每天都有该功能; 若为其他, 根据设置开关: 位 0: 周一; 若为 1, 表示每周一都有该功能; 位 1: 周二; 若为 1, 表示每周二都有该功能;

位 2: 周三; 若为 1, 表示每周三都有该功能;
位 3: 周四; 若为 1, 表示每周四都有该功能;
位 4: 周五; 若为 1, 表示每周五都有该功能;
位 5: 周六; 若为 1, 表示每周六都有该功能;
位 6: 周日; 若为 1, 表示每周日都有该功能;
位 7: 保留, 始终为 0;

on_hour [IN] 指向存储开机时钟的内存空间。

on_minute [IN] 指向存储开机分钟的内存空间。

off_hour [IN] 指向存储关机时钟的内存空间。

off_minute [IN] 指向存储关机分钟的内存空间。

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能描述:

查询自动开关机信息。

127. JHCreateBatchFile1 (创建空批处理文件)

```
JHZRESULT JHZAPI JHCreateBatchFile1(  
    HANDLE * phdl,  
    const char * pformatstring)
```

参数:

phdl [IN] 指向存储实例句柄的内存空间。

pformatstring [IN] 指向存储批处理文件名格式化字符串的内存空间。

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

IR_NOT_ENOUGH_CORE 内存不足

功能描述:

创建空的批处理文件 里面不包含批处理文件数据内容, 此后关于批处理的操作都是基于handle参数;此handle参数只是保存批处理数据对它一切操作, 只有当所有批处理文件数据准备好后才会与控制卡通讯。

128. JHCreateBatchFile2 (创建批处理文件)

```
JHZRESULT JHZAPI JHCreateBatchFile2(  
    HANDLE * phdl,  
    const char * pformatstring,
```

```
unsigned char * pdat,  
unsigned long size)
```

参数:

phdl [IN] 指向存储实例句柄的内存空间。
pformatstring [IN] 指向存储批处理文件名格式化字符串的内存空间。
pdat [IN] 指向存储批处理文件数据的内存空间。
size [IN] 指向存储批处理文件数据大小的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在已知批处理文件数据的情况下 向phdl指向的实例中填充批处理数据, 以下对批处理文件数据的操作都是基于此phdl参数。

129. JHDeleteBatchFile (删除批处理文件)

```
HZRESULT JHZAPI JHDeleteBatchFile(  
HANDLE phdl)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。

返回值:

JR_OK	成功
-------	----

功能描述:

从实例句柄对应的控制卡中删除删除批处理文件。

130. JHGetBatchFileData (获取批处理文件数据)

```
JHZRESULT JHZAPI JHGetBatchFileData(  
HANDLE phdl,  
unsigned char * pdat)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
pdat [OUT] 指向存储批处理文件数据的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

从phdl参数对应的实例中获得批处理文件数据。

131. JHGetBatchFileSize（获取批处理文件大小）

```
JHZRESULT JHZAPI JHGetBatchFileSize(  
    HANDLE phdl,  
    unsigned long * psize)
```

参数:

phdl [IN] 实例句柄，与控制卡对应。
psize [OUT] 指向存储批处理文件数据大小的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

从phdl参数对应的实例中获取批处理文件的大小。

132. JHGetBatchFileName（获取批处理文件名）

```
JHZRESULT JHZAPI JHGetBatchFileName(  
    HANDLE phdl,  
    char * pformatstring)
```

参数:

phdl [IN] 实例句柄，与控制卡对应。
pformatstring [OUT] 指向存储批处理文件名格式化字符串的内存空间。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

从phdl对应的实例中获取批处理文件名。

133. JHSetBatchFileName（设置批处理文件名）

```
JHZRESULT JHZAPI JHSetBatchFileName(  
    HANDLE phdl,  
    char * pformatstring)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
pformatstring [IN] 指向存储批处理文件名格式化字符串的内存空间。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
IR_INVALID_OBJECT 无效的对象

功能描述:

设置phdl对应的实例中的批处理文件名。

134. JHSetCurLumIst (设置当前亮度指令)

```
JHZRESULT JHZAPI JHSetCurLumIst(  
    HANDLE phdl,  
    unsigned short index,  
    unsigned short lum)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
lum [IN] 显示屏当前亮度值, 有效值为 0x0000-0x03DE。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
IR_INVALID_OBJECT 无效的对象
IR_NOT_ENOUGH_CORE 内存不足

功能描述:

在批处理文件中索引为index的位置上添加设置当前亮度指令。该指令设置的亮度掉电后不生效。

135. JHSetCurFontIst (设置当前字体指令)

```
JHZRESULT JHZAPI JHSetCurFontIst(  
    HANDLE phdl,  
    unsigned short index,  
    unsigned char fid)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
fid [IN] 字体 ID, 有效值为 1-254。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为index的位置上添加设置当前字体指令。该指令设置的字体掉电后不生效。

136. JHSetCurPenColorIst（设置当前字体颜色指令）

```
JHZRESULT JHZAPI JHSetCurPenColorIst(
    HANDLE phdl,
    unsigned short index,
    unsigned long color)
```

参数：

phdl [IN] 实例句柄，与控制卡对应。

index [IN] 指令在批处理文件中基于 1 开始的索引。

color [IN] 字体颜色，颜色定义如下：

- 位 7-4：保留，始终为 0；
- 位 3-0：颜色（注：不能为 0000；黑色为 1000）
- 位 2：蓝；
- 位 1：绿；
- 位 0：红

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为index的位置上添加设置当前字体颜色指令。该指令设置的字体颜色掉电后不生效。

137. JHSetCurBrushColorIst（设置当前画刷颜色指令）

```
JHZRESULT JHZAPI JHSetCurBrushColorIst(
    HANDLE phdl,
    unsigned short index,
    unsigned long color)
```

参数：

phdl [IN] 实例句柄，与控制卡对应。

index [IN] 指令在批处理文件中基于 1 开始的索引。

color [IN] 画刷颜色，颜色定义如下：

- 位 7-4: 保留，始终为 0;
- 位 3-0: 颜色（注：不能为 0000；黑色为 1000）
- 位 2: 蓝;
- 位 1: 绿;
- 位 0: 红

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加设置当前画刷颜色指令。该指令设置的画刷颜色掉电后不生效。

138. JHPowerIst（开关机指令）

```
JHZRESULT JHZAPI JHPowerIst(
    HANDLE phdl,
    unsigned short index,
    unsigned char onoff)
```

参数:

phdl [IN] 实例句柄，与控制卡对应。

index [IN] 指令在批处理文件中基于 1 开始的索引。

onoff [IN] 开机状态，0x00 表示关机，其他表示开机。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加开关机指令。该指令执行后，控制卡系统会根据onoff的值做开机或关机的操作。

139. JHRestartIst（重启指令）

```
JHZRESULT JHZAPI JHRestartIst(
```



```
HANDLE phdl,  
unsigned short index)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加重启指令。重启指令执行后, 控制卡会重新上电。

140. JHEraseIst (清屏指令)

```
JHZRESULT JHZAPI JHEraseIst(  
HANDLE phdl,  
unsigned short index)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加清屏指令。该指令执行后会清除整个屏幕的显示内容。

141. JHDrawMultiPixelIst (点亮点组指令)

```
JHZRESULT JHZAPI JHDrawMultiPixelIst(  
HANDLE phdl,  
unsigned short index,  
const char * pdat,  
unsigned short poscnt)
```

参数:

phdl [IN] 实例句柄，与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
pdat [IN] 指向点组数据的指针。
poscnt [IN] 点组总数。

返回值：

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
IR_INVALID_OBJECT 无效的对象
IR_NOT_ENOUGH_CORE 内存不足

功能描述：**添加**第index（从1开始）条（**点亮点组指令**）批处理指令到包含批处理数据的handle 里。

142. JHDrawLineIst（画线指令）

```
JHZRESULT JHZAPI JHDrawLineIst(  
    HANDLE phdl,  
    unsigned short index,  
    short x1,  
    short y1,  
    short x2,  
    short y2)
```

参数：

phdl [IN] 实例句柄，与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
x1 [IN] 直线起始点 X 坐标。
y1 [IN] 直线起始点 Y 坐标。
x2 [IN] 直线结束点 X 坐标。
y2 [IN] 直线结束点 Y 坐标。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为index的位置上添加画线指令。直线的起始坐标为(x1, y1)，结束坐标为(x2, y2)，直线的颜色由当前画刷确定。画刷颜色可以由设置当前画刷颜色指令JHSetCurBrushColorIst修改。

143. JHFillRectIst（填充矩形指令）

```
JHZRESULT JHZAPI JHFillRectIst(  
    HANDLE phdl,  
    unsigned short index,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height)
```

参数:

phdl	[IN] 实例句柄，与控制卡对应。
Index	[IN] 指令在批处理文件中基于 1 开始的索引。
x	[IN] 矩形左上角起始点 X 坐标。
y	[IN] 矩形左上角起始点 Y 坐标。
width	[IN] 矩形宽度。
height	[IN] 矩形高度。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加填充矩形指令。矩形的区域由 x, y, width, height 确定，填充的颜色为当前画刷的颜色。画刷颜色可以由设置当前画刷颜色指令 JHSetCurBrushColorIst 修改。

144. JHDrawTextIst（立即显示文本指令）

```
JHZRESULT JHZAPI JHDrawTextIst(  
    HANDLE phdl,  
    unsigned short index,  
    short x,  
    short y,  
    unsigned short width,  
    unsigned short height,  
    unsigned long format,  
    const char * pformatstring)
```

参数:

phdl	[IN] 实例句柄，与控制卡对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。

x	[IN] 文本显示位置左上角的 X 坐标。
y	[IN] 文本显示位置左上角的 Y 坐标。
width	[IN] 文本显示区域宽度，为 0x0000 时为自适应。
height	[IN] 文本显示区域高度，为 0x0000 时为自适应。
format	[IN] 文本显示格式。 位 31-20：保留，始终为 0。 位 19- 16：颜色（注：0000 表示使用系统当前颜色；1000 为黑色） 位 18：蓝色。 位 17：绿色。 位 16：红色。 位 15- 8：字体 ID；有效值为 0-254；为 0 时使用当前字体 ID。 位 7- 6：水平对齐 HALIGN 00：左对齐。 01：水平居中对齐。 10：右对齐。 位 5- 4：垂直对齐方式 00：上对齐。 01：垂直居中对齐。 10：下对齐。 位 3：保留，始终为 0。 位 2：wordbreak，自动换行，为 0，不自动换行，为 1，自动换行； 位 1：单行文本； 0 表示多行文本，1 表示单行文本。 位 0：高级文本，0 表示普通文本，1 表示高级文本；保留，始终为 0。
pformatstring	[IN] 指向存储待显示的文本字符串的内存空间。文本字符编码，可包含有效的 ASCII 码(不能包含 0x00)和 GB2312 编码，可混用。换行符为' \n'，即 0x0A；水平制表符' \t'，即 0x09，显示为 4 个 ASCII 空格符。

返回值

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为index的位置上添加显示文本指令。显示的文本内容为 pformatstring指向的字符串，显示的区域为x, y, width, height确定的矩形内。

145. JHCreateWndIst（创建分区指令）

JHZRESULT JHZAPI JHCreateWndIst(

```

HANDLE phdl,
unsigned short index,
unsigned short wid,
unsigned short option,
unsigned short x,
unsigned short y,
unsigned short width,
unsigned short height,
unsigned long style,
unsigned long userdata)

```

参数:

phdl	[IN] 实例句柄，与控制卡对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号，有效值为 0x0001-0x0003。
option	[IN] 创建方式，0x0000 表示创建时若分区编号已存在，则创建失败；为 0x0001 时，表示不管是否已存在，均创建(若已存在，则覆盖，覆盖时，会先清除分区原有节目)。
x	[IN] 分区左上角起始点 X 坐标。
y	[IN] 分区左上角起始点 Y 坐标。
width	[IN] 分区宽度。
height	[IN] 分区高度。
style	[IN] 分区样式。 位 31- 3: 保留，始终为 0; 位 1: 永久; 0 为临时分区, 1 为永久分区; 位 0: 使能; 保留, 始终为 0;
userdata	[IN] 用户数据，保留，始终为 0。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加创建分区指令。被创建的分区编号为wid, 分区左上角起始点X坐标为x, 分区左上角起始点Y坐标为y, 分区宽度为width, 分区高度为height。

146. JHChangeWndIst（改变分区属性指令）

```

JHZRESULT JHZAPI JHChangeWndIst(
    HANDLE phdl,
    unsigned short index,

```

unsigned short wid,
unsigned short x,
unsigned short y,
unsigned short width,
unsigned short height,
unsigned long style,
unsigned long userdata)

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
wid [IN] 分区编号, 有效值为 0x0001-0x0003。
x [IN] 分区左上角起始点 X 坐标。
y [IN] 分区左上角起始点 Y 坐标。
width [IN] 分区宽度。
height [IN] 分区高度。
style [IN] 分区样式。
位 31- 3: 保留, 始终为 0;
位 1: 永久; 0 为临时分区, 1 为永久分区;
位 0: 使能; 保留, 始终为 0;
userdata [IN] 用户数据, 保留, 始终为 0。

返回值:

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
IR_INVALID_OBJECT 无效的对象
IR_NOT_ENOUGH_CORE 内存不足

功能描述:

在批处理文件中索引为index的位置上添加改变分区属性指令。被修改的分区编号为wid, 该分区的左上角起始点X坐标修改为x, 左上角起始点Y坐标修改为y, 分区宽度修改为width, 分区高度修改为height。

147. JHDeleteWndIst (删除分区指令)

JHZRESULT JHZAPI JHDeleteWndIst(
HANDLE phdl,
unsigned short index,
unsigned short wid)

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
wid [IN] 分区编号, 有效值为 0x0001-0x0003, 或 0xFFFF; 为 0xFFFF 时为删除所有分区。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为 index 的位置上添加删除分区指令，删除的分区编号为 wid。

148. JHCreateTextProgIst（创建文本节目指令）

```
JHZRESULT JHZAPI JHCreateTextProgIst(
    HANDLE phdl,
    unsigned short index,
    unsigned short wid,
    unsigned short rev,
    unsigned long style,
    unsigned long format,
    const JHTEXTPROG * ptextprog,
    const char * pformatstring)
```

参数：

phdl	[IN] 实例句柄，与控制卡对应。
Index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号，有效值为 0x0001-0x0003，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63； 位 7-0: 保留，始终为 0。
style	[IN] 节目样式。 位 31-4: 保留，始终为 0； 位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）； 位 0: 保留，始终为 0；
format	[IN] 文本显示格式。 位 31-20: 保留，始终为 0。 位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色） 位 18: 蓝色。 位 17: 绿色。 位 16: 红色。

位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。

位 7- 6: 水平对齐 HALIGN

00: 左对齐。

01: 水平居中对齐。

10: 右对齐。

位 5- 4: 垂直对齐方式

00: 上对齐。

01: 垂直居中对齐。

10: 下对齐。

位 3: 保留, 始终为 0。

位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;

位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。

位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。

ptextprog

[IN] 指向 JHTEXTPROG 类型的内存空间。JHTEXTPROG 结构的格式如下所示:

tpEntryMode: 进入效果; 0x0000 不合法, 强制转为 0x0001。

0x0001: 立即显示

0x0002: 向左移入

0x0003: 向右移入

0x0004: 向上移入

0x0005: 向下移入

0x0006: 向右展开

0x0007: 向左展开

0x0008: 向上展开

0x0009: 向下展开

0x000A: 左右合并

0x000B: 左右展开

0x000C: 上下合并

0x000D: 画卷右展开

0x000E: 画卷左展开

0x000F: 画卷左右展开

0x0010: 画卷左右合并

0x0011: 画卷上下合并

0x0012: 拉开窗帘

0x0013: 中间溢出

0x0014: 左右移动

0x0015: 右下抽出

0x0016: 右上抽出

0x0017: 左下抽出

0x0018: 左上抽出

0x0019: 水平百叶窗

0x001A: 垂直百叶窗

0x001B:矩形扩散
0x001C:矩形收缩
0x001D:菱形扩散
0x001E:菱形收缩
0x001F:十字扩散
0x0020:顺时针 1 根轮辐
0x0021:顺时针 2 根轮辐
0x0022:顺时针 4 根轮辐
0x0023:顶端画扇形
0x0024:底端画扇形
0x0025:中间画扇形
0x0026:四顶点画扇形
0x0027:水平梳理
0x0028:垂直梳理
0x0029:左右穿插
0x002A:飘雪
0x002B:冒泡
0x002C:左镭射
0x002D:右镭射
0x002E:上下生长
0x002F:左开始波形
0x0030:右开始波形
0xFFFF:随机进入

tpEntrySpeed: 进入效果速度; 该值越大, 运动速度越慢。

tpEntryStaytime: 进入效果停留时间; 单位 10 毫秒。

tpHighLightMode: 0x0000 时无强调效果, 0x0001 时闪烁;

tpHighLightSpeed: 有强调效果时生效, 闪烁的速度。

tpHighLightStaytime: 强调效果停留时间; 单位 10 毫秒。

tpExitMode: 退出效果。

0x0000:无退出效果
0x0001:立即清屏
0x0002:向左移出
0x0003:向右移出
0x0004:向上移出
0x0005:向下移出
0x0006:左右合并
0x0007:左右展开
0x0008:上下合并
0xFFFF:随机退出

tpExitSpeed: 退出效果速度; 该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。

pformatstring [IN] 指向存储待显示的文本字符串的内存空间。文本字符编码, 可包含有效的 ASCII 码(不能包含 0x00)和 GB2312 编码, 可混用。换行

符为' \n' ,即 0x0A; 水平制表符' \t' ,即 0x09,显示为 4 个 ASCII 空格符。

注: FORMAT 属性说明, 主要注意两类, 单行文本和多行文本

1. 单行文本时:

忽略 wordbreak 自动换行设置, 无论是自动换行或是手动在命令中添加换行符' \n' , 均忽略, 不作换行处理,

2. 多行文本时:

根据设置不作限制, 自动换行和手动换行可同时支持;

但是若文本显示所需区域超过当前限制区域, 建议做如下处理:

1. 若为单行文本, 建议不要设置水平居中; 其他可任意搭配;

2. 若为多行文本, 若使用了自动换行, 建议不要使用垂直居中对齐; 若无自动换行, 建议不要使用任何居中对齐方式;

使用时若遇到上面情况, 可参照上面说明使用, 否则为了整体显示效果, 系统会自行裁剪部分数据;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为 index 的位置上添加创建文本节目指令。将要创建的文本在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域, 每个区域以一个分区编号来标记。

149. JHCreateSimpleDigClockProgIst(创建简易时钟指令)

```
JHZRESULT JHZAPI JHCreateSimpleDigClockProgIst(  
    HANDLE phdl,  
    unsigned short index,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset)
```

参数:

phdl	[IN] 实例句柄, 与控制卡对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号, 有效值为 0x0001-0x0003, 或 0xFFFF; 为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识, 用于单分区多节目。

	位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63;
	位 7-0: 保留, 始终为 0。
style	[IN] 节目样式。
	位 31-4: 保留, 始终为 0;
	位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载;
	位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除;
	位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失);
	位 0: 保留, 始终为 0;
format	[IN] 文本显示格式。
	位 31-20: 保留, 始终为 0。
	位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色)
	位 18: 蓝色。
	位 17: 绿色。
	位 16: 红色。
	位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。
	位 7- 6: 水平对齐 HALIGN
	00: 左对齐。
	01: 水平居中对齐。
	10: 右对齐。
	位 5- 4: 垂直对齐方式。
	00: 上对齐。
	01: 垂直居中对齐。
	10: 下对齐。
	位 3: 保留, 始终为 0。
	位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
	位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。
	位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。
timeoffset	[IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。

注: FORMAT 属性说明, 主要注意两类, 单行文本和多行文本

1. 单行文本时:

忽略 wordbreak 自动换行设置, 无论是自动换行或是手动在命令中添加换行符' \n' , 均忽略, 不作换行处理,

2. 多行文本时:

根据设置不作限制, 自动换行和手动换行可同时支持;

但是若文本显示所需区域超过当前限制区域, 建议做如下处理:

1. 若为单行文本, 建议不要设置水平居中; 其他可任意搭配;

2. 若为多行文本, 若使用了自动换行, 建议不要使用垂直居中对齐; 若无自动换行, 建议不要使用任何居中对齐方式;

使用时若遇到上面情况, 可参照上面说明使用, 否则为了整体显示效果, 系统会自行裁剪部

分数据：

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为 index 的位置上添加创建简易时钟指令。将要创建的简易时钟在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域，每个区域以一个分区编号来标记。简易时钟的显示格式是默认的，用户不能修改。如果用户需要指定时钟的显示格式，可以使用 JHCreateDigClockProgIst（创建高级时钟指令）。

150. JHCreateDigClockProgIst（创建数字时钟指令）

```
JHZRESULT JHZAPI JHCreateDigClockProgIst(  
    HANDLE phdl,  
    unsigned short index,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset,  
    const char * pformatstring)
```

参数：

phdl	[IN] 实例句柄，与控制卡对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号，有效值为 0x0001-0x0003，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63； 位 7-0: 保留，始终为 0。
style	[IN] 节目样式。 位 31-4: 保留，始终为 0； 位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载； 位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除； 位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）； 位 0: 保留，始终为 0；
format	[IN] 文本显示格式。

位 31-20: 保留, 始终为 0。

位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色)

位 18: 蓝色。

位 17: 绿色。

位 16: 红色。

位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。

位 7- 6: 水平对齐 HALIGN

00: 左对齐。

01: 水平居中对齐。

10: 右对齐。

位 5- 4: 垂直对齐方式

00: 上对齐。

01: 垂直居中对齐。

10: 下对齐。

位 3: 保留, 始终为 0。

位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;

位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。

位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。

timeoffset [IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。

pformatstring [IN] 指向存储文本格式化字符串的内存空间。文本格式化字符串的格式如下 (注意, 下面描述中注意大小写的形式, 若须要显示 “%”, 则在 sFORMAT 中需要填补两个百分号, 即 “%%”; 换行符为 ‘\n’, 即 0x0A; 水平制表符 ‘\t’, 即 0x09, 显示为 4 个 ASCII 空格符):

年: %yyyy 或 %yy, 分别用于显示四位年份或两位年份;

月: %M 或 %MM; %M 为月份是几位就显示几位, %MM 为不足两位数的月份, 前面补零;

日: %d 或 %dd; %d 为日期是几位就显示几位, %dd 为不足两位数的日期, 前面补零;

时: %H, %HH, %h, %hh; H 是 24 小时制, h 是 12 小时制; %H 和 %h 为时钟是几位就显示几位, %HH 和 %hh 为不足两位数的时钟, 前面补零;

分: %m 或 %mm; %m 为分钟是几位就显示几位, %mm 为不足两位数的分钟, 前面补零;

秒: %s 或 %ss; %s 为秒钟是几位就显示几位, %ss 为不足两位数的秒钟, 前面补零;

星期: %w 或 %W; %w 为英文星期缩写; %W 为中文星期数;

%w: Sun; Mon; Tue; Wed; Thu; Fri; Sat;

%W: 星期日; 星期一; 星期二; 星期三; 星期四; 星期五; 星期六;

上/下午: %t 或 %T; %t 为英文, AM 或 PM; %T 为中文, 上午或下午;

一个文本格式化字符串可以为 “%yyyy 年 %MM 月 %dd 日 %HH 时 %mm 分 %ss 秒 %W”, 年、月、日、时、分、秒、星期都是可选的, 例如, 如果不

需要在显示屏上显示年份，文本格式化字符串改为“%MM 月%dd 日 %HH 时%mm 分%ss 秒 %W”。

注：FORMAT 属性说明，主要注意两类，单行文本和多行文本

1. 单行文本时：

忽略 wordbreak 自动换行设置，无论是自动换行或是手动在命令中添加换行符‘\n’，均忽略，不作换行处理，

2. 多行文本时：

根据设置不作限制，自动换行和手动换行可同时支持；

但是若文本显示所需区域超过当前限制区域，建议做如下处理：

1. 若为单行文本，建议不要设置水平居中；其他可任意搭配；

2. 若为多行文本，若使用了自动换行，建议不要使用垂直居中对齐；若无自动换行，建议不要使用任何居中对齐方式；

使用时若遇到上面情况，可参照上面说明使用，否则为了整体显示效果，系统会自行裁剪部分数据；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

在批处理文件中索引为 index 的位置上添加创建数字时钟指令。将要创建的高级时钟在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域，每个区域以一个分区编号来标记。创建高级时钟需要用户指定时钟的显示格式，用户也可以使用 JHCreateSimpleDigClockProgIst（创建简易时钟指令）创建默认格式显示的时钟。

151. JHCreateComplexDigClockProgIst（创建高级数字时钟指令）

```
JHZRESULT JHZAPI JHCreateComplexDigClockProg(  
    HANDLE hdl,  
    unsigned short index,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    unsigned long timeoffset,  
    const JHTEXTPROG * ptextprog,
```

const char * pformatstring);

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号, 有效值为控制卡中已经创建的分区, 或 0xFFFF; 为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识, 用于单分区多节目。 位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63; 位 7-0: 保留, 始终为 0。
style	[IN] 节目样式。 位 31-4: 保留, 始终为 0; 位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载; 位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除; 位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失); 位 0: 保留, 始终为 0;
format	[IN] 文本显示格式。 位 31-20: 保留, 始终为 0。 位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色) 位 18: 蓝色。 位 17: 绿色。 位 16: 红色。 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。 位 7- 6: 水平对齐 HALIGN 00: 左对齐。 01: 水平居中对齐。 10: 右对齐。 位 5- 4: 垂直对齐方式 00: 上对齐。 01: 垂直居中对齐。 10: 下对齐。 位 3: 保留, 始终为 0。 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行; 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。
timeoffset	[IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。
ptextprog	[IN] 指向存储 JHTEXTPROG 结构数据的内存空间, 结构成员如下: tpEntryMode: 进入效果; 0x0000 不合法, 强制转为 0x0001. 0x0001:立即显示 0x0002:向左移入 0x0003:向右移入

0x0004: 向上移入
0x0005: 向下移入
0x0006: 向右展开
0x0007: 向左展开
0x0008: 向上展开
0x0009: 向下展开
0x000A: 左右合并
0x000B: 左右展开
0x000C: 上下合并
0x000D: 画卷右展开
0x000E: 画卷左展开
0x000F: 画卷左右展开
0x0010: 画卷左右合并
0x0011: 画卷上下合并
0x0012: 拉开窗帘
0x0013: 中间溢出
0x0014: 左右移动
0x0015: 右下抽出
0x0016: 右上抽出
0x0017: 左下抽出
0x0018: 左上抽出
0x0019: 水平百叶窗
0x001A: 垂直百叶窗
0x001B: 矩形扩散
0x001C: 矩形收缩
0x001D: 菱形扩散
0x001E: 菱形收缩
0x001F: 十字扩散
0x0020: 顺时针 1 根轮辐
0x0021: 顺时针 2 根轮辐
0x0022: 顺时针 4 根轮辐
0x0023: 顶端画扇形
0x0024: 底端画扇形
0x0025: 中间画扇形
0x0026: 四顶点画扇形
0x0027: 水平梳理
0x0028: 垂直梳理
0x0029: 左右穿插
0x002A: 飘雪
0x002B: 冒泡
0x002C: 左镭射
0x002D: 右镭射
0x002E: 上下生长
0x002F: 左开始波形

0x0030:右开始波形

0xFFFF:随机进入

tpEntrySpeed: 进入效果速度; 该值越大, 运动速度越慢;

tpEntryStaytime: 进入效果停留时间; 单位 10 毫秒;

tpHighLightMode: 0x0000 时无强调效果, 0x0001 时闪烁;

tpHighLightSpeed: 有强调效果时生效, 闪烁的速度。

tpHighLightStaytime: 强调效果停留时间; 单位 10 毫秒。

tpExitMode: 退出效果。

0x0000:无退出效果

0x0001:立即清屏

0x0002:向左移出

0x0003:向右移出

0x0004:向上移出

0x0005:向下移出

0x0006:左右合并

0x0007:左右展开

0x0008:上下合并

0xFFFF:随机退出

tpExitSpeed: 退出效果速度; 该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。

pformatstring [IN] 指向存储文本格式化字符串的内存空间。文本格式化字符串的格式如下(注意, 下面描述中注意大小写的形式, 若须要显示 “%”, 则在 sFORMAT 中需要填补两个百分号, 即 “%%”; 换行符为 ‘\n’, 即 0x0A; 水平制表符 ‘\t’, 即 0x09, 显示为 4 个 ASCII 空格符):

年: %yyyy 或 %yy, 分别用于显示四位年份或两位年份;

月: %M 或 %MM; %M 为月份是几位就显示几位, %MM 为不足两位数的月份, 前面补零;

日: %d 或 %dd; %d 为日期是几位就显示几位, %dd 为不足两位数的日期, 前面补零;

时: %H, %HH, %h, %hh; H 是 24 小时制, h 是 12 小时制; %H 和 %h 为时钟是几位就显示几位, %HH 和 %hh 为不足两位数的时钟, 前面补零;

分: %m 或 %mm; %m 为分钟是几位就显示几位, %mm 为不足两位数的分钟, 前面补零;

秒: %s 或 %ss; %s 为秒钟是几位就显示几位, %ss 为不足两位数的秒钟, 前面补零;

星期: %w 或 %W; %w 为英文星期缩写; %W 为中文星期数;

%w: Sun; Mon; Tue; Wed; Thu; Fri; Sat;

%W: 星期日; 星期一; 星期二; 星期三; 星期四; 星期五; 星期六;

上/下午: %t 或 %T; %t 为英文, AM 或 PM; %T 为中文, 上午或下午;

一个文本格式化字符串可以为 “%yyyy 年 %MM 月 %dd 日 %HH 时 %mm 分 %ss

秒 %W”，年、月、日、时、分、秒、星期都是可选的，例如，如果不需要在显示屏上显示年份，文本格式化字符串改为“%MM 月%dd 日 %HH 时%mm 分%ss 秒 %W”。

注：FORMAT 属性说明，主要注意两类，单行文本和多行文本

1. 单行文本时：

忽略 wordbreak 自动换行设置，无论是自动换行或是手动在命令中添加换行符‘\n’，均忽略，不作换行处理，

2. 多行文本时：

根据设置不作限制，自动换行和手动换行可同时支持；

但是若文本显示所需区域超过当前限制区域，建议做如下处理：

1. 若为单行文本，建议不要设置水平居中；其他可任意搭配；

2. 若为多行文本，若使用了自动换行，建议不要使用垂直居中对齐；若无自动换行，建议不要使用任何居中对齐方式；

使用时若遇到上面情况，可参照上面说明使用，否则为了整体显示效果，系统会自行裁剪部分数据；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在批处理文件中索引为 index 的位置上添加创建高级时钟指令。将要创建的高级时钟在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域，每个区域以一个分区编号来标记。创建高级时钟需要用户指定时钟的显示格式，用户也可以使用 JHCreateSimpleDigClockProgIst（创建简易时钟指令）创建默认格式显示的时钟。

152. JHCreateBitmapProgIst（创建图片节目指令）

```
JHZRESULT JHZAPI JHCreateBitmapProgIst(  
    HANDLE phdl,  
    unsigned short index,  
    unsigned short wid,  
    unsigned short rev,  
    unsigned long style,  
    unsigned long format,  
    const JHTEXTPROG * ptextprog,  
    unsigned short rev2,  
    const JHBMPSRC * pbmpsrc);
```

参数：

phdl	[IN] 实例句柄，与控制卡对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF

时为向所有分区发送同一个节目。

rev [IN] 节目标识，用于单分区多节目。
 位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63；
 位 7-0: 保留，始终为 0。

style [IN] 节目样式。
 位 31-4: 保留，始终为 0；
 位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载；
 位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除；
 位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）；
 位 0: 保留，始终为 0；

Format [IN] 图片显示格式。
 位 31-8: 保留，为 0；
 位 7- 6: 水平对齐 HALIGN
 00: 左对齐；
 01: 水平居中对齐；
 10: 右对齐；
 位 5- 4:
 00: 上对齐；
 01: 垂直居中对齐；
 10: 下对齐；
 位 3-0: 保留，为 0；

ptextprog [IN] 指向 JHTEXTPROG 类型的内存空间。JHTEXTPROG 结构体格式如下所示：

tpEntryMode: 进入效果。
 0x0001: 立即显示。

tpEntrySpeed: 进入效果速度。该值越大，运动速度越慢。

tpEntryStaytime: 进入效果停留时间，单位 10 毫秒。

tpHighLightMode: 强调。
 0x0000: 无强调效果
 0x0001: 闪烁

tpHighLightSpeed: 强调效果速度。该值越大，运动速度越慢。

tpHighLightStaytime: 强调效果停留时间，单位 10 毫秒。

tpExitMode: 退出效果。
 0x0000: 无退出效果
 0x0001: 立即清屏
 0x0002: 向左移出
 0x0003: 向右移出
 0x0004: 向上移出
 0x0005: 向下移出
 0x0006: 左右合并
 0x0007: 左右展开
 0x0008: 上下合并

0xFFFF:随机退出

tpExitSpeed: 退出效果速度。该值越大, 运动速度越慢。

tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。

rev2 [IN] 保留; 始终为 0。

JHBMPSRC [IN] 图片源信息。

```
struct tagBitmapSource{
    unsigned short type;
    unsigned short version;
    unsigned long size;
    unsigned char *pdata;
} JHBMPSRC;

type: 图片类型, 始终为 0, 表示为文件图片;
version: 版本, 当前版本为 0;
size: 图片源数据大小;
pdata: 图片源数据; data 为动态数组, 大小由 size 决定;
```

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为 index 的位置上添加创建图片节目指令。将要创建的图片节目在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域, 每个区域以一个分区编号来标记。

153. JHCreateQRCodeProgIst (创建二维码节目指令)

```
JHZRESULT JHZAPI JHCreateQRCodeProgIst(
    HANDLE phdl,
    unsigned short index,
    unsigned short wid,
    unsigned short rev,
    unsigned long style,
    unsigned long format,
    const JHTEXTPROG * ptextprog,
    const JHQRCODE* pqrcode,
    const char * pformatstring)
```

参数:

phdl	[IN] 实例句柄, 与控制卡对应。
Index	[IN] 指令在批处理文件中基于 1 开始的索引。
wid	[IN] 分区编号, 有效值为 0x0001-0x0003, 或 0xFFFF; 为 0xFFFF

	时为向所有分区发送同一个节目。
rev	[IN] 节目标识，用于单分区多节目。 位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63; 位 7-0: 保留, 始终为 0。
style	[IN] 节目样式。 位 31-4: 保留, 始终为 0; 位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载; 位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除; 位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失); 位 0: 保留, 始终为 0;
format	[IN] 文本显示格式。 位 31-20: 保留, 始终为 0。 位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色) 位 18: 蓝色。 位 17: 绿色。 位 16: 红色。 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。 位 7- 6: 水平对齐 HALIGN 00: 左对齐。 01: 水平居中对齐。 10: 右对齐。 位 5- 4: 垂直对齐方式 00: 上对齐。 01: 垂直居中对齐。 10: 下对齐。 位 3: 保留, 始终为 0。
ptextprog	[IN] 指向 JHTEXTPROG 类型的内存空间。JHTEXTPROG 结构的格式如下所示: tpEntryMode: 进入效果; 0x0000 不合法, 强制转为 0x0001。 0x0001: 立即显示 0x0002: 向左移入 0x0003: 向右移入 0x0004: 向上移入 0x0005: 向下移入 0x0006: 向右展开 0x0007: 向左展开 0x0008: 向上展开 0x0009: 向下展开 0x000A: 左右合并 0x000B: 左右展开

0x000C:上下合并
0x000D:画卷右展开
0x000E:画卷左展开
0x000F:画卷左右展开
0x0010:画卷左右合并
0x0011:画卷上下合并
0x0012:拉开窗帘
0x0013:中间溢出
0x0014:左右移动
0x0015:右下抽出
0x0016:右上抽出
0x0017:左下抽出
0x0018:左上抽出
0x0019:水平百叶窗
0x001A:垂直百叶窗
0x001B:矩形扩散
0x001C:矩形收缩
0x001D:菱形扩散
0x001E:菱形收缩
0x001F:十字扩散
0x0020:顺时针 1 根轮辐
0x0021:顺时针 2 根轮辐
0x0022:顺时针 4 根轮辐
0x0023:顶端画扇形
0x0024:底端画扇形
0x0025:中间画扇形
0x0026:四顶点画扇形
0x0027:水平梳理
0x0028:垂直梳理
0x0029:左右穿插
0x002A:飘雪
0x002B:冒泡
0x002C:左镭射
0x002D:右镭射
0x002E:上下生长
0x002F:左开始波形
0x0030:右开始波形
0xFFFF:随机进入

tpEntrySpeed: 进入效果速度; 该值越大, 运动速度越慢。

tpEntryStaytime: 进入效果停留时间; 单位 10 毫秒。

tpHighLightMode: 0x0000 时无强调效果, 0x0001 时闪烁;

tpHighLightSpeed: 有强调效果时生效, 闪烁的速度。

tpHighLightStaytime: 强调效果停留时间; 单位 10 毫秒。

tpExitMode: 退出效果。

0x0000:无退出效果
 0x0001:立即清屏
 0x0002:向左移出
 0x0003:向右移出
 0x0004:向上移出
 0x0005:向下移出
 0x0006:左右合并
 0x0007:左右展开
 0x0008:上下合并
 0xFFFF:随机退出
 tpExitSpeed: 退出效果速度; 该值越大, 运动速度越慢。
 tpRepeatTimes: 重复次数, 有效值为 0x001-0xFFFF; 0xFFFF 表示除非有新的节目过来, 否则一直循环播放。
 Pqrcode [IN] 指向存储 JHQRCODE 结构数据的内存空间, 结构成员如下:

```

typedef struct{
    unsigned short rev;
    unsigned short type;
    unsigned short version;
}JHQRCODE;
rev :保留, 始终为 0。
type : 类型, 二维码图片节目值为 2 (位图图片节目值为 0)。
version: 保留, 始终为 0。

```

 pformatstring [IN] 指向存储待显示的二维码文本字符串的内存空间 (1-128 字节)。文本字符编码, 可包含有效的 ASCII 码 (不能包含 0x00)。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为 index 的位置上添加创建二维码节目指令。将要创建的维码在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域, 每个区域以一个分区编号来标记。

154. JHCreateCountDownProgIst (创建倒计时指令)

```

JHZRESULT JHZAPI JHCreateCountDownProgIst(
    HANDLE phdl,
    unsigned short index,
    unsigned short wid,
    unsigned short rev,

```

```

unsigned long style,
unsigned long format,
unsigned long timeoffset,
unsigned long flag,
const JHSYSTEMTIME * ptime,
const char * pformatstring);

```

参数:

- phdl [IN] 实例句柄, 与控制卡对应。
- index [IN] 指令在批处理文件中基于 1 开始的索引。
- wid [IN] 分区编号, 有效值为控制卡中已经创建的分区, 或 0xFFFF; 为 0xFFFF 时为向所有分区发送同一个节目。
- rev [IN] 节目标识, 用于单分区多节目。
 位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63;
 位 7-0: 保留, 始终为 0。
- style [IN] 节目样式。
 位 31-4: 保留, 始终为 0;
 位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载;
 位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除;
 位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失);
 位 0: 保留, 始终为 0;
- format [IN] 文本显示格式。
 位 31-20: 保留, 始终为 0。
 位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色)
 位 18: 蓝色。
 位 17: 绿色。
 位 16: 红色。
 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID。
 位 7- 6: 水平对齐 HALIGN
 00: 左对齐。
 01: 水平居中对齐。
 10: 右对齐。
 位 5- 4: 垂直对齐方式
 00: 上对齐。
 01: 垂直居中对齐。
 10: 下对齐。
 位 3: 保留, 始终为 0。
 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行;
 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本。
 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。
- timeoffset [IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。
- flag [IN] 控制信息

位 31-1: 保留, 必须为 0;

位 0: 参考时间类型, 为 0 表示 ptime 为绝对时间, 为 1 表示 ptime 为相对当前的时间;

注: 当 FLAG 的位 0 为 1 时, ptime 中只有 wDay、wHour、wMinute、wSecond 有效, 参考的时间点为加载时的时间和这四个值所表示的时间之和 ((wDay*86400) + (wHour*3600) + (wMinute*60) + wSecond) 。

ptime [IN] 指向 JHSYSTEMTIME 类型的内存空间。JHSYSTEMTIME 结构的格式如下所示:

wYear: 年

wMonth: 月, 有效值为 1-12;

wDayOfWeek: 星期, 保留值, 始终为 0

wDay: 日, 有效值为 1-31;

wHour: 时, 有效值为 0-23;

wMinute: 分, 有效值为 0-59;

wSecond: 秒, 有效值为 0-59;

wMilliseconds: 毫秒, 保留值, 始终为 0;

注: 表示截止时间 (倒计时到何时结束), 其类型由 FLAG 的位 0 决定。

pformatstring [IN] 指向存储待显示的时间格式字符串内存空间。时间格式化字符串格式如下:

不能包含 0x00; (注意, 下面描述中注意大小写的形式)

日: %D, 总的天数;

时: %H 或 %h, %H 为总的时钟数, 有几位就显示几位; %h 为当前天中的时钟数, 不足两位数的时钟, 前面补零;

分: %M 或 %m; %M 为总的分钟数, 有几位就显示几位; %m 为当前小时中的分钟数, 不足两位数的分钟, 前面补零;

秒: %S 或 %s; %S 为总的秒钟数, 有几位就显示几位; %s 为当前分钟中的秒数, 不足两位数的秒钟, 前面补零;

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

IR_INVALID_OBJECT 无效的对象

IR_NOT_ENOUGH_CORE 内存不足

功能描述:

在批处理文件中索引为 index 的位置上添加创建倒计时指令。将要创建的倒计时在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域, 每个区域以一个分区编号来标记。创建倒计时用户可指定倒计时的显示格式。

155. JHCreateCountUpProgIst (创建正计时指令)

```
JHZRESULT JHZAPI JHCreateCountUpProgIst(  
    HANDLE phdl,
```

```

unsigned short index,
unsigned short wid,
unsigned short rev,
unsigned long style,
unsigned long format,
unsigned long timeoffset,
unsigned long flag,
const JHSYSTEMTIME * ptime,
const char * pformatstring);

```

参数:

- | | |
|--------|---|
| phdl | [IN] 实例句柄，与控制卡对应。 |
| index | [IN] 指令在批处理文件中基于 1 开始的索引。 |
| wid | [IN] 分区编号，有效值为控制卡中已经创建的分区，或 0xFFFF；为 0xFFFF 时为向所有分区发送同一个节目。 |
| rev | [IN] 节目标识，用于单分区多节目。
位 15-8: 节目编号，同分区中具有唯一性，值越小播放顺序越靠前，有效范围 0~63；
位 7-0: 保留，始终为 0。 |
| style | [IN] 节目样式。
位 31-4: 保留，始终为 0；
位 3: 是否按队列加载，仅永久节目有效，为 0 表示立即，为 1 表示按队列顺序加载；
位 2: 加载时是否擦除分区，为 0 表示不擦除，为 1 表示擦除；
位 1: 节目属性，为 0 表示临时节目（即掉电易失），为 1 表示永久节目（掉电不易失）；
位 0: 保留，始终为 0； |
| format | [IN] 文本显示格式。
位 31-20: 保留，始终为 0。
位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色）
位 18: 蓝色。
位 17: 绿色。
位 16: 红色。
位 15- 8: 字体 ID；有效值为 0-254；为 0 时使用当前字体 ID。
位 7- 6: 水平对齐 HALIGN
00: 左对齐。
01: 水平居中对齐。
10: 右对齐。
位 5- 4: 垂直对齐方式
00: 上对齐。
01: 垂直居中对齐。
10: 下对齐。
位 3: 保留，始终为 0。
位 2: wordbreak，自动换行，为 0，不自动换行，为 1，自动换行；
位 1: 单行文本； 0 表示多行文本，1 表示单行文本。 |

位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0。

timeoffset [IN] 时间偏移, 单位秒, 可用于实现多时区时间显示, 有符号数值。

flag [IN] 控制信息

位 31-1: 保留, 必须为 0;

位 0: 参考时间类型, 为 0 表示 ptime 为绝对时间, 为 1 表示 ptime 为相对当前的时间;

注: 当 FLAG 的位 0 为 1 时, ptime 中只有 wDay、wHour、wMinute、wSecond 有效, 参考的时间点为加载时的时间和这四个值所表示的时间之和 ($(wDay*86400) + (wHour*3600) + (wMinute*60) + wSecond$)。

ptime [IN] 指向 JHSYSTEMTIME 类型的内存空间。JHSYSTEMTIME 结构的格式如下所示:

wYear: 年

wMonth: 月, 有效值为 1-12;

wDayOfWeek: 星期, 保留值, 始终为 0

wDay: 日, 有效值为 1-31;

wHour: 时, 有效值为 0-23;

wMinute: 分, 有效值为 0-59;

wSecond: 秒, 有效值为 0-59;

wMilliseconds: 毫秒, 保留值, 始终为 0;

注: ptime 表示起始时间 (从何时开始计时), 其类型由 FLAG 的位 0 决定; 若将其设置为 NULL 则表示 ptime 的所有值为 0。

pformatstring [IN] 指向存储待显示的时间格式字符串内存空间。时间格式化字符串格式如下:

不能包含 0x00; (注意, 下面描述中注意大小写的形式)

日: %D, 总的天数;

时: %H 或 %h, %H 为总的时钟数, 有几位就显示几位; %h 为当前天中的时钟数, 不足两位数的时钟, 前面补零;

分: %M 或 %m; %M 为总的分钟数, 有几位就显示几位; %m 为当前小时中的分钟数, 不足两位数的分钟, 前面补零;

秒: %S 或 %s; %S 为总的秒钟数, 有几位就显示几位; %s 为当前分钟中的秒数, 不足两位数的秒钟, 前面补零;

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

IR_INVALID_OBJECT 无效的对象

IR_NOT_ENOUGH_CORE 内存不足

功能描述:

在批处理文件中索引为 index 的位置上添加创建正计时指令。将要创建的正计时在分区编号为 wid 的显示区域内显示。一个显示屏可以分为若干个区域, 每个区域以一个分区编号来标记。创建正计时用户可指定正计时的显示格式。

156. JHMoveIst（移动指令）

```
JHZRESULT JHZAPI JHMoveIst(  
    HANDLE phdl,  
    unsigned short oindex,  
    unsigned short nindex)
```

参数：

phdl [IN] 实例句柄，与控制卡对应。
oindex [IN] 指令移动前在批处理文件中基于 1 开始的索引。
nindex [IN] 指令移动后在批处理文件中基于 1 开始的索引。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

移动批处理指令的顺序，从索引 oindex 的位置上移动到索引 nindex 的位置上。

157. JHDeleteIst（删除指令）

```
JHZRESULT JHZAPI JHDeleteIst(  
    HANDLE phdl,  
    unsigned short index)
```

参数：

phdl [IN] 实例句柄，与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述：

删除批处理文件中索引为 index 的指令。

158. JHGetIstCnt（获取指令总数）

```
JHZRESULT JHZAPI JHGetIstCnt(  
    HANDLE phdl,
```

```
unsigned short * pCnt)
```

参数:

phdl [IN]实例句柄, 与控制卡对应。

pCnt [OUT]指向 unsigned short 类型的内存空间, 该内存空间存储批处理文件中指令的总数。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

获取批处理文件中的指令总数, 批处理文件中可包含多个指令。

159. JHGetIstCmd (获取指令 CMD)

```
JHZRESULT JHZAPI JHGetIstCmd(  
    HANDLE phdl,  
    unsigned short index,  
    unsigned short * pcmd)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。

index [IN] 指令在批处理文件中基于 1 开始的索引。

pcmd [OUT]指向 unsigned short 类型的内存空间, 该内存空间存储批处理文件中索引为 index 的指令对应的 cmd 编码。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

获取批处理指令的 cmd 编码, 如 0x0038 为设置当前亮度的 cmd 编码。不同功能对应不同的 cmd, 可在文档《ZkLED 字库卡 v2.x 使用手册》中查询对应关系。

160. JHGetIstData (获取指令数据)

```
JHZRESULT JHZAPI JHGetIstData(  
    HANDLE phdl,  
    unsigned short index,  
    char * pdat,  
    unsigned short * psize)
```

参数:

phdl [IN]实例句柄, 与控制卡对应。
index [IN]指令在批处理文件中基于 1 开始的索引。
pdat [OUT]指向存储指令数据的内存空间。
psize [OUT]指向 unsigned short 类型的内存空间, 该内存空间存储指令数据的长度。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

获得批处理文件中索引为index的批处理指令数据, 指令数据格式请参照文档《ZkLED字库卡v2.x使用手册》。

161. JHSetIstData (设置指令数据)

```
JHZRESULT JHZAPI JHSetIstData(  
    HANDLE phdl,  
    unsigned short index,  
    const char * pdat,  
    unsigned short size)
```

参数:

phdl [IN] 实例句柄, 与控制卡对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
pdat [IN] 指向存储指令数据的内存空间。
size [IN] 指令数据的长度, 不能小于 0。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象
IR_NOT_ENOUGH_CORE	内存不足

功能描述:

设置批处理文件中索引为index的批处理指令数据, 指令数据格式请参照文档《ZkLED字库卡v2.x使用手册》。

162. JHCreateGridProg (创建表格节目)

```
JHZRESULT JHZAPI JHCreateGridProg(
    HANDLE hdl,
    unsigned short wid,
    unsigned short rev,
    unsigned long style,
    unsigned long format,
    unsigned long flag,
    const char * pformatstring)
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
wid	[IN] 分区编号, 有效值为 0x0001-0x0003, 或 0xFFFF; 为 0xFFFF 时为向所有分区发送同一个节目。
rev	[IN] 节目标识, 用于单分区多节目。 位 15-8: 节目编号, 同分区中具有唯一性, 值越小播放顺序越靠前, 有效范围 0~63; 位 7-0: 保留, 始终为 0。
style	[IN] 节目样式。 位 31-4: 保留, 始终为 0; 位 3: 是否按队列加载, 仅永久节目有效, 为 0 表示立即, 为 1 表示按队列顺序加载; 位 2: 加载时是否擦除分区, 为 0 表示不擦除, 为 1 表示擦除; 位 1: 节目属性, 为 0 表示临时节目 (即掉电易失), 为 1 表示永久节目 (掉电不易失); 位 0: 保留, 始终为 0;
format	[IN] 文本显示格式。 位 31-20: 保留, 始终为 0; 位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色) 位 18: 蓝色; 位 17: 绿色; 位 16: 红色; 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID; 位 7- 6: 水平对齐 HALIGN 00: 左对齐; 01: 水平居中对齐; 10: 右对齐; 位 5- 4: 00: 上对齐; 01: 垂直居中对齐; 10: 下对齐; 位 3: 保留, 始终为 0; 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行; 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本; 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留, 始终为 0;

注意：format 不要超过 1200 字节。

flag [IN] 控制信息
位 31-16: 节目停留时间（单位：秒），仅在位 1 为 0 时有效；
位 15-2: 保留；
位 1: 为 0 表示节目时间无限长，为 1 表示有限节目时间；
注意：若设置为 0 则不能和其他表格节目切换显示，只显示当前节目。
位 0: 为 0 表示加载节目时擦除当前分区，为 1 表示加载节目时不擦除当前分区；
pformatstring [IN] 指向存储待显示的表格格式字符串的内存空间。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
JH_INVALID_OBJECT	无效的对象

功能描述：

在实例句柄对应控制卡上，根据指定的参数在分区编号为 wid 的分区上创建表格节目。

163. JHGridCreateHandle（创建表格实例句柄）

JHZRESULT JHZAPI JHGridCreateHandle(
HANDLE* phdl)

参数：

phdl [IN] 实例句柄，与表格相对应。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不足

功能描述：

创建一个和实例句柄对应的表格的具体实例。

164. JHGridSetRowColCount（设置表格行数和列数）

JHZRESULT JHZAPI JHGridSetRowColCount(
HANDLE phdl,


```
    unsigned short rowCount,  
    unsigned short colCount)
```

参数:

phdl	[IN] 实例句柄, 与表格相对应。
rowCount	[IN] 表格行数
colCount	[IN] 表格列数

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

设置实例句柄对应表格的行数和列数。

165. JHGridSetDefRowHeightColWidth (设置表格默认行高和列宽)

```
JHZRESULT JHZAPI JHGridSetDefRowHeightColWidth(  
    HANDLE phdl,  
    unsigned short height,  
    unsigned short width)
```

参数:

phdl	[IN] 实例句柄, 与表格相对应。
height	[IN] 高度。
width	[IN] 宽度。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

设置实例句柄对应表格默认行高和列宽。

166. JHGridSetDefTextFormat (设置默认表格文本格式)

```
JHZRESULT JHZAPI JHGridSetDefTextFormat(  
    HANDLE phdl,
```

unsigned long format)

参数:

phdl	[IN] 实例句柄，与表格相对应。
format	[IN] 文本显示格式。 位 31-20: 保留，始终为 0; 位 19- 16: 颜色 (注: 0000 表示使用系统当前颜色; 1000 为黑色) 位 18: 蓝色; 位 17: 绿色; 位 16: 红色; 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID; 位 7- 6: 水平对齐 HALIGN 00: 左对齐; 01: 水平居中对齐; 10: 右对齐; 位 5- 4: 00: 上对齐; 01: 垂直居中对齐; 10: 下对齐; 位 3: 保留，始终为 0; 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行; 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本; 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留，始终为 0。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

表格中文本格式默认按照 format 参数确定的格式显示文本。

167. JHGridSetRowHeight (设置表格指定行行高)

```
JHZRESULT JHZAPI JHGridSetRowHeight(  
    HANDLE phdl,  
    unsigned short rowNum,  
    unsigned short height)
```

参数:

phdl	[IN] 实例句柄，与表格相对应。
rowNum	[IN] 指定行号 (从0开始)。
height	[IN] 高度。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

设置实例句柄对应表格第 rowNum 行的行高。

168. JHGridSetColWidth （设置表格指定列列宽）

```
JHZRESULT JHZAPI JHGridSetColWidth(
    HANDLE phdl,
    unsigned short colNum,
    unsigned short width)
```

参数:

Phdl	[IN] 实例句柄，与表格相对应。
colNum	[IN] 指定列号（从0开始）。
width	[IN] 宽度。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能描述:

设置实例句柄对应表格第colNum列的列宽。

169. JHGridSetCellText （设置单元格文本）

```
JHZRESULT JHZAPI JHGridSetCellText(
    HANDLE phdl,
    unsigned short row,
    unsigned short col,
    const char* ptext)
```

参数:

phdl	[IN] 实例句柄，与表格相对应。
row	[IN] 单元格所在行号。
col	[IN] 单元格所在列号。
ptext	[IN] 指向存储待显示的文本字符串的内存空间。文本字符编码，可包含有效的ASCII码(不能包含0x00)和GB2312编码，可混用。 换

行符为' \n' ，即0x0A；水平制表符' \t' ，即0x09，显示为4个ASCII空格符。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

设置实例句柄对应表格第 row 行 col 列的单元格文本。

170. JHGridInsertRowsAt （插入行）

```
JHZRESULT JHZAPI JHGridInsertRowsAt(  
    HANDLE phdl,  
    unsigned short rowNum,  
    unsigned short count)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
rowNum	[IN] 行号（从0开始）。
count	[IN] 插入行数。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

在实例句柄对应表格的rowNum行下插入count行。

171. JHGridInsertColsAt (插入列)

```
JHZRESULT JHZAPI JHGridInsertColsAt(  
    HANDLE phdl,  
    unsigned short colNum,  
    unsigned short count)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
colNum	[IN] 列号（从0开始）。
count	[IN] 插入列数。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

在实例句柄对应表格的第colNum列后插入count列。

172. JHGridRemoveColsAt （删除列）

```
JHZRESULT JHZAPI JHGridRemoveColsAt(
    HANDLE phdl,
    unsigned short colNum,
    unsigned short count)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
colNum	[IN] 列号（从0开始）。
count	[IN] 删除的列数。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

将实例句柄对应表格的第colNum列删除count列，包含第colNum列。

173. JHGridRemoveRowsAt （删除行）

```
JHZRESULT JHZAPI JHGridRemoveRowsAt(
    HANDLE phdl,
    unsigned short rowNum,
    unsigned short count)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
rowNum	[IN] 行号（从0开始）。
count	[IN] 删除的行数。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明:

将实例句柄对应表格的第rowNum行删除count行, 包含第rowNum行。

174. JHGridMergeCells (合并单元格)

```
JHZRESULT JHZAPI JHGridMergeCells(  
    HANDLE phdl,  
    unsigned short startRow,  
    unsigned short startCol,  
    unsigned short endRow,  
    unsigned short endCol)
```

参数:

phdl	[IN] 实例句柄, 与表格相对应。
startRow	[IN] 起始单元格行号 (从0开始)。
startCol	[IN] 起始单元格列号 (从0开始)。
endRow	[IN] 终止单元格行号 (从0开始)。
endCol	[IN] 终止单元格列号 (从0开始)。

注1: endRow 和endCol 设置都不能包含其本身所在行列数。比如对一个2*4的表格要合并其第2行第2列和第2行第3列两个单元格则可以这样设置: JHGridMergeCells(phdl, 1, 1, 2, 3);

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明:

将实例句柄对应表格的第startRow行startCol列到第endRow行endCol列的所有单元格合并。

175. JHGridSetCellTextFormat (设置单元格文本格式)

```
JHZRESULT JHZAPI JHGridSetCellTextFormat(  
    HANDLE phdl,  
    unsigned short row,  
    unsigned short col,  
    unsigned long format)
```

参数:

phdl	[IN] 实例句柄, 与表格相对应。
------	--------------------

row	[IN] 行号（从0开始）。
col	[IN] 列号（从0开始）。
format	[IN] 文本显示格式。 位 31-20: 保留，始终为 0; 位 19- 16: 颜色（注：0000 表示使用系统当前颜色；1000 为黑色） 位 18: 蓝色; 位 17: 绿色; 位 16: 红色; 位 15- 8: 字体 ID; 有效值为 0-254; 为 0 时使用当前字体 ID; 位 7- 6: 水平对齐 HALIGN 00: 左对齐; 01: 水平居中对齐; 10: 右对齐; 位 5- 4: 00: 上对齐; 01: 垂直居中对齐; 10: 下对齐; 位 3: 保留，始终为 0; 位 2: wordbreak, 自动换行, 为 0, 不自动换行, 为 1, 自动换行; 位 1: 单行文本; 0 表示多行文本, 1 表示单行文本; 位 0: 高级文本, 0 表示普通文本, 1 表示高级文本; 保留，始终为 0。

返回值:

JH_OK	成功
JH_ INVALID_PARAMETER	参数错误
IR_ INVALID_OBJECT	无效的对象

功能说明:

设置实例句柄对应表格第row行col列的单元格文本格式。

176. JHGridSetBorders (设置单元格边框)

```
JHZRESULT JHZAPI JHGridSetBorders(
    HANDLE phdl,
    unsigned short row,
    unsigned short col,
    unsigned char border)
```

参数:

Phdl	[IN] 实例句柄，与表格相对应。
row	[IN] 行号（从0开始）。
col	[IN] 列号（从0开始）。
border	[IN] 边框属性。

注:边框属性

Ctrl1	Ctrl2
border高4位	border低4位

Ctrl1 和 Ctrl2 的字符取值范围为 0-9、A-F、a-f。

Ctrl1 转为 16 进制数范围为 0x0—0xf，有 4 个比特位：表示边框颜色：

（注：0000 表示使用系统当前颜色；1000 为黑色）

位 2：蓝色；

位 1：绿色；

位 0：红色；

Ctrl2 转为 16 进制数范围为 0x0—0xf：表示边框样式：

0x0： 1 像素宽度直线（默认）

0x1： 2 像素宽度直线；

其他值：保留；

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

设置实例句柄对应表格第row行col列的单元格边框属性。

177. JHGridGetCellText (获取单元格文本内容)

```
JHZRESULT JHZAPI JHGridGetCellText(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned char * pdat,  
    unsigned short* pSize)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
usRow	[IN] 行号（从0开始）。
usCol	[IN] 列号（从0开始）。
pdat	[OUT] 指向待存储单元格文本字符串的内存空间。
pSize	[OUT] pdat字符串大小。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

获取实例句柄对应表格第usRow行usCol列的单元格文本字符串内容及大小。

178. JHGridGetRowCount (获取表格行数)

```
JHZRESULT JHZAPI JHGridGetRowCount(  
    HANDLE phdl,  
    unsigned short * pCount)
```

参数:

phdl	[IN] 实例句柄, 与表格相对应。
pCount	[OUT] 指向存储表格行数的内存空间。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明:

获取实例句柄对应表格的行数。

179. JHGridGetColCount (获取表格列数)

```
JHZRESULT JHZAPI JHGridGetColCount(  
    HANDLE phdl,  
    unsigned short * pCount)
```

参数:

phdl	[IN] 实例句柄, 与表格相对应。
pCount	[OUT] 指向存储表格列数的内存空间。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明:

获取实例句柄对应表格的列数。

180. JHGridGetRowHeight (获取表格某一行行高)

```
JHZRESULT JHZAPI JHGridGetRowHeight(  
    HANDLE phdl,  
    unsigned short rowNum,  
    unsigned short* height)
```

参数:

phdl	[IN] 实例句柄，与表格相对应。
rowNum	[IN] 行号（从0开始）。
height	[OUT] 指向存储表格行高的内存空间。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

获取实例句柄对应表格第rowNum行的行高。

181. JHGridGetColWidth (获取表格某一列列宽)

```
JHZRESULT JHZAPI JHGridGetColWidth(  
    HANDLE phdl,  
    unsigned short colNum,  
    unsigned short* width)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
colNum	[IN] 列号（从0开始）。
width	[OUT] 指向存储表格列宽的内存空间。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

获取实例句柄对应表格第colNum列的列宽。

182. JHGridIsMergeCell （获取单元格是否合并信息）

```
JHZRESULT JHZAPI JHGridIsMergeCell(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned char * pbMerge)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
usRow	[IN] 单元格所在行号（从0开始）。
usCol	[IN] 单元格所在列号（从0开始）。

pbMerge [OUT] 指向存储单元格是否在合并区域的字符的内存空间。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明:

判断实例句柄对应表格的第usRow行usCol列单元格是否是处在合并的单元格区域。

183. JHGridGetMergeArea (获取合并区域)

```
JHZRESULT JHZAPI JHGridGetMergeArea(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned short* pstartRow,  
    unsigned short* pstartCol,  
    unsigned short* pendRow,  
    unsigned short* pendCol)
```

参数:

phdl	[IN] 实例句柄，与表格相对应。
usRow	[IN] 单元格所在行号（从0开始）。
usCol	[IN] 单元格所在列号（从0开始）。
pstartRow	[OUT] 指向合并区域起始单元格所在行号的内存空间。
pstartCol	[OUT] 指向合并区域起始单元格所在列号的内存空间。
pendRow	[OUT] 指向合并区域终止单元格所在行号的内存空间。
pendCol	[OUT] 指向合并区域终止单元格所在列号的内存空间。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明:

获得实例句柄对应表格的第usRow行usCol列单元格的合并区域。

184. JHGridGetBorders (获取单元格边框)

```
JHZRESULT JHZAPI JHGridGetBorders(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned char * pborders)
```

参数:

phdl [IN] 实例句柄，与表格相对应。
usRow [IN] 单元格所在行号（从0开始）。
usCol [IN] 单元格所在列号（从0开始）。
pborders [OUT] 指向存储单元格边框属性的格式化字符串的内存空间。

注:边框属性

Ctrl1	Ctrl2
pborder高4位	pborder低4位

Ctrl1 和 Ctrl2 的字符取值范围为 0-9、A-F、a-f。
Ctrl1 转为 16 进制数范围为 0x0—0xf，有 4 个比特位:表示边框颜色：
(注：0000 表示使用系统当前颜色；1000 为黑色)

位 2: 蓝色；
位 1: 绿色；
位 0: 红色；

Ctrl2 转为 16 进制数范围为 0x0—0xf:表示边框样式：
0x0: 1 像素宽度直线(默认)
0x1: 2 像素宽度直线；
其他值: 保留；

返回值:

JH_OK 成功
JH_INVALID_PARAMETER 参数错误
JR_INVALID_OBJECT 无效的对象

功能说明:

获取实例句柄对应表格的边框属性。

185. JHGridGetCellTextFormat (获取单元格文本格式)

```
JHZRESULT JHZAPI JHGridGetCellTextFormat(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned long * pformat)
```

参数:

phdl [IN] 实例句柄，与表格相对应。
usRow [IN] 单元格所在行号（从0开始）。
usCol [IN] 单元格所在列号（从0开始）。
pformat [OUT] 指向存储单元格文本格式的内存空间。

返回值:

JH_OK 成功
JH_INVALID_PARAMETER 参数错误
JR_INVALID_OBJECT 无效的对象

功能说明：

获取实例句柄对应表格的第usRow行usCol列单元格中文本的格式。

186. JHGridDeleteHandle (删除表格实例句柄)

```
JHZRESULT JHZAPI JHGridDeleteHandle(  
    HANDLE phdl)
```

参数：

phdl [IN] 实例句柄，与表格相对应。

返回值：

JH_OK 成功

JR_INVALID_OBJECT 无效的对象

功能说明：

通过指定实例句柄指针来删除已经创建好的实例。该api将删除phdl指向的实例，删除后释放保存phdl对应表格相关信息的内存空间。

187. JHGridFormatToHandle （表格显示格式化文本转为表格句柄）

```
JHZRESULT JHZAPI JHGridFormatToHandle(  
    HANDLE phdl,  
    const char * pformatstring)
```

参数：

phdl [IN] 实例句柄，与表格相对应。

pformatstring [IN] 指向存储表格文本格式字符串的内存空间。

返回值：

JH_OK 成功

JH_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能说明：

将存储表格的格式化文本字符串转换为表格句柄。

188. JHGridFormatToString (表格句柄转为表格显示格式化文本)

```
JHZRESULT JHZAPI JHGridFormatToString(  
    HANDLE phdl,  
    char * pdat,  
    unsigned short bufsize,  
    unsigned short* size)
```

参数:

phdl	[IN] 实例句柄，与表格相对应。
pdat	[OUT] 指向存储文本格式字符串的内存空间。
bufsize	[IN] 缓冲区表格字符串大小。
size	[OUT] 获取到的表格字符串大小。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能说明:

将表格的实例句柄phdl转化为格式化文本字符串存储在pdat指向的内存空间。

189. JHGridSetSlash (设置斜线)

```
JHZRESULT JHZAPI JHGridSetSlash(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned char bSlash)
```

参数:

phdl	[IN] 实例句柄，与表格相对应。
usRow	[IN] 单元格所在行号（从0开始）。
usCol	[IN] 单元格所在列号（从0开始）。
bSlash	[IN] 斜线类型设置标志。

注: bBackSlash为非0值 : 表示设置斜线“\”; 为0表示不设置斜线“\”。

返回值:

JH_OK 成功

其他情况，请查阅错误类型对照表。

功能说明:

将实例句柄对应表格的第usRow行usCol列单元格设置或不设置斜线“\”。

190. JHGridGetSlash (获取斜线)

```
JHZRESULT JHZAPI JHGridGetSlash(
    HANDLE phdl,
    unsigned short usRow,
    unsigned short usCol,
    unsigned char* pbSlash)
```

参数:

phd1	[IN] 实例句柄，与表格相对应。
usRow	[IN] 单元格所在行号（从0开始）。
usCol	[IN] 单元格所在列号（从0开始）。
bSlash	[OUT] 指向存储单元格是否可设置反斜线的字符串的内存空间。

返回值:

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能说明:

获得实例句柄对应表格的第usRow行usCol列单元格是否设置了斜线，bBackSlash 为非0值则表示此单元格设置了反斜线，为0表示未设置。

191. JHGridSetBackSlash (设置反斜线)

```
JHZRESULT JHZAPI JHGridSetBackSlash(
    HANDLE phdl,
    unsigned short usRow,
    unsigned short usCol,
    unsigned char bBackSlash)
```

参数:

phd1	[IN] 实例句柄，与表格相对应。
usRow	[IN] 单元格所在行号（从0开始）。
usCol	[IN] 单元格所在列号（从0开始）。
bBackSlash	[IN] 反斜线类型设置标志。

注：bBackSlash为非0值：表示设置反斜线“/”；为0表示不设置反斜线“/”。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

将实例句柄对应表格的第usRow行usCol列单元格设置或不设置反斜线“/”。

192. JHGridGetBackSlash (获取反斜线)

```
JHZRESULT JHZAPI JHGridGetBackSlash(  
    HANDLE phdl,  
    unsigned short usRow,  
    unsigned short usCol,  
    unsigned char* pbBackSlash)
```

参数：

phdl	[IN] 实例句柄，与表格相对应。
usRow	[IN] 单元格所在行号（从0开始）。
usCol	[IN] 单元格所在列号（从0开始）。
bBackSlash	[OUT] 指向存储反斜线类型的文本字符串的内存空间。

返回值：

JH_OK	成功
JH_INVALID_PARAMETER	参数错误
IR_INVALID_OBJECT	无效的对象

功能说明：

获得实例句柄对应表格的第usRow行usCol列单元格是否设置了反斜线，bBackSlash为非0值则表示此单元格设置了反斜线，为0表示未设置。

193. JHGetBitmapCnt (查询图片数目)

```
JHZRESULT JHZAPI JHGetBitmapCnt(  
    HANDLE hdl,  
    unsigned short * pcnt);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
pcnt	[OUT] 指向存储图片数目的内存空间。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡里图片的数目。

194. JHGetBitmapId(查询图片 ID)

```
JHZRESULT JHZAPI JHGetBitmapId(
    HANDLE hdl,
    unsigned short index,
    unsigned short * pbid);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
index	[IN] 图片序号；系统中该序号为连续序号，区分不同的图片；该值从 0 开始。
pbid	[OUT] 指向存储图片 ID 的内存空间，图片 ID 的有效值为 1-65534，，用于标识图片文件。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡中指定序号的图片 ID，序号从 0 开始，图片 ID 的有效值 1-65534。

195. JHGetBitmapInfo(查询图片信息)

```
JHZRESULT JHZAPI JHGetBitmapInfo(
    HANDLE hdl,
    unsigned short bid,
    BITMAPINFOHEADER * pbitmap);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
fid	[IN] 字体 ID，有效值为 1-65534，用于标识字库文件。
Pbitmap	[OUT] 指向 BITMAPINFOHEADER 类型的内存空间，BITMAPINFOHEADER 结构体如下所示： <div> biSize: 本结构所占用字节数； biWidth: 图片的宽度，以像素为单位； biHeight: 图片的高度，以像素为单位； biPlanes: 目标设备的级别，必须为 1； </div>

biBitCount: 每个像素所需的位数;
biCompression: 图片压缩类型, BI_RGB(0) 为不压缩;
biSizeImage: 图片的大小 (单位: 字节), BI_RGB 时可设为 0;
biXPelsPerMeter: 图片水平分辨率 (像素/米);
biYPelsPerMeter: 图片垂直分辨率 (像素/米);
biClrUsed: 图片实际使用的颜色表中的颜色数, 如果为 0, 则颜色数 2 的 biBitCount 次方;
biClrImportant: 图片显示过程中重要的颜色数, 0 代表所有颜色都重要;

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

获取实例句柄对应控制卡中图片 ID 号为 bid 的图片信息。

196. JHInquireBitmap (查询图片是否存在)

```
JHZRESULT JHZAPI JHInquireBitmap(  
    HANDLE hdl,  
    unsigned short bid);
```

参数:

hdl [IN] 实例句柄, 与控制卡相对应。
bid [IN] 图片 ID, 有效值为 1-65534, 用于标识图片文件。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

查询实例句柄对应控制卡中图片 ID 为 bid 的字库是否存在, 返回 JR_OK 表示存在, 否则为不存在。

197. JHPreCreateBitmap (下载图片数据第一步发送图片数据总大小)

```
JHZRESULT JHZAPI JHPreCreateBitmap(  
    HANDLE hdl,  
    HANDLE *bd,  
    unsigned short bid,
```

```
unsigned long writesize);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
bd	[OUT] 指向发送图片第一步中申请的存储图片下载操作过程信息的内存空间, 发送图片第二步需要根据这个信息进行操作, 第三步会释放 bd 指向的内存空间。
bid	[IN] 图片 ID, 有效值为 1-65534, 用于标识图片文件。
writesize	[IN] 图片数据总大小。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不够

功能描述:

发送图片数据操作的第一步, 申请一块内存保存下载操作过程的信息, 提供给发送图片操作第二步使用。

198. JHWriteBitmap (下载图片数据第二步发送图片数据)

```
JHZRESULT JHZAPI JHWriteBitmap(  
    HANDLE hdl,  
    HANDLE bd,  
    unsigned long woffset,  
    const char * pdat,  
    unsigned long size);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
bd	[IN] 指向发送位图第一步中申请的存储图片下载操作过程信息的内存空间, 发送图片第二步需要根据这个信息进行操作, 第三步会释放 fd 指向的内存空间。
woffset	[IN] 本次所要发送数据在图片数据中的偏移。
pdat	[IN] 指向存储本次所要发送数据的内存空间。
size	[IN] 本次所要发送数据的大小, 不超过 1200 字节, 不少于 1 个字节。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

发送图片数据到实例句柄对应的控制卡上, 当图片内容较大时, 需分包发送。

使用示例:

```
HANDLE hdl, bd;  
char[4000] dat={1};  
int num, bid;
```

```

    bid = 1;
    JHPreCreateBitmap (hdl, &hd, bid ,size);
    pdat = dat;
    num = 0;
    if(size > 1200)
    {
        while(num < size)
        {
            JHWriteBitmap(hdl,bd,pdat-dat,pdat,num+1200<size ? 1200 : size-num) ;
            num+=1200;
        }
    }
    JHFinCreateBitmap(hdl, bd);

```

199. JHFinCreateBitmap（下载图片数据第三步结束下载，释放资源）

```

JHZRESULT JHZAPI JHFinCreateBitmap(
    HANDLE hdl,
    HANDLE bd);

```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
 bd [IN] 指向发送图片第一步中申请的存储图片下载操作过程信息的内存空间，发送图片第二步需要根据这个信息进行操作，第三步会释放 fd 指向的内存空间。

返回值：

JR_OK 成功
 JR_INVALID_OBJECT 无效的对象

功能描述：

通知实例句柄对应控制卡图片数据发送完成，释放bd资源。

200. JHDeleteBitmap（删除图片文件）

```

JHZRESULT JHZAPI JHDeleteBitmap(
    HANDLE hdl,
    unsigned short bid);

```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
 bid [IN] 图片 ID，有效值为 1-65534，用于标识图片文件。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

删除实例句柄对应控制卡中图片 ID 为 bid 的图片文件。

201. JHSpeechSynthesisStart（开始语音合成）

```
JHZRESULT JHZAPI JHSpeechSynthesisStart(  
    HANDLE hdl,  
    unsigned long option,  
    unsigned short num,  
    const char *pformatstring);
```

参数:

hdl	[IN] 实例句柄，与控制卡相对应。
option	[IN] 操作选项。 位 31-17: 保留; 位 17: 0 不重复, 1 自动重复; 位 16: 是否等待合成，为 0 表示立即语音合成，为 1 表示等待上一次语音合成结束后再合成本次语音; 位 15-8: 背景音乐，0 表示无背景音乐，1-15 表示背景音乐编号; 位 7-0: 字符编码; 0x00: GB2312 编码; 0x01: GBK 编码; 0x02: BIG5 编码; 0x03: UNICODE 编码; 其他保留;
num	[IN] 字符个数，1-1024 字节;（单位：字节）。
Pformatstring	[IN] 要语音合成的文本。

返回值:

JR_OK	成功
JR_INVALID_OBJECT	无效的对象

功能描述:

用于开始播放合成的语音文本。

202. JHSpeechSynthesisStop（停止语音合成）

```
JHZRESULT JHZAPI JHSpeechSynthesisStop(  
    HANDLE hd1);
```

参数:

hd1 [IN] 实例句柄，与控制卡相对应。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

用于终止播放合成的语音文本。

203. JHSpeechSynthesisPause（暂停语音合成）

```
JHZRESULT JHZAPI JHSpeechSynthesisPause(  
    HANDLE hd1);
```

参数:

hd1 [IN] 实例句柄，与控制卡相对应。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

用于暂停播放合成的语音文本。

204. JHSpeechSynthesisRestart（恢复语音合成）

```
JHZRESULT JHZAPI JHSpeechSynthesisRestart(  
    HANDLE hd1);
```

参数:

hd1 [IN] 实例句柄，与控制卡相对应。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

用于恢复暂停播放的语音文本。

205. JHSpeechSynthesisInquire（查询语音模块）

```
JHZRESULT JHZAPI JHSpeechSynthesisInquire(  
    HANDLE hdl ,  
    unsigned char* pstat);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
pstat [OUT] 指向存储当前语音状态的内存空间。
 位 7-1：保留，为 0；
 位 0：为 0 表示空闲，为 1 表示正在工作中；

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

用于查询当前语音的播放状态。

206. JHSpeechSynthesisSleep（休眠语音模块）

```
JHZRESULT JHZAPI JHSpeechSynthesisSleep(  
    HANDLE hdl);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

用于让语音模块处于休眠状态。

注意：如果已经调用 **JHSpeechSynthesisSleep**（休眠语音模块）指令，当再次调用时会返回
错误代码：58；故请勿重复调用！

207. JHSpeechSynthesisWakeup（唤醒语音模块）

```
JHZRESULT JHZAPI JHSpeechSynthesisWakeup(  
    HANDLE hdl);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

用于唤醒休眠的语音模块。

208. JHSpeechSynthesisSetPara（设置语音参数）

```
JHZRESULT JHZAPI JHSpeechSynthesisSetPara (  
    HANDLE hdl  
    unsigned short para,  
    unsigned short rev0,  
    unsigned short rev1,  
    unsigned short rev2,  
);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
para	[IN] 修改语音音量参数。
rev0	[IN] 保留。
Rev1	[IN] 保留。
Rev2	[IN] 保留。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

用于设置语音参数，目前只支持修改音量。

209. JHSetIOState（设置 IO 输出状态）

```
JHZRESULT JHZAPI JHSetIOState (  
    HANDLE hdl  
    unsigned char ch,  
    unsigned char state,  
);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
-----	--------------------

ch [IN] IO 通道号，有效值为 0-7，不同的硬件可能会有差异。
state [IN] 输出状态，0 输出低电平(0V)，1 输出高电平(3.3V)。

返回值：

JR_OK 成功
JR_INVALID_PARAMETER 参数错误
JR_INVALID_OBJECT 无效的对象

功能描述：

用于设置 IO 的输出状态。

注意：IO 引脚最大输出电流约为 15mA，连接到外接设备时，要串联 330 欧姆电阻，防止损坏字库卡。

210. JHDrawBitmapIst (立即显示图片指令)

```
JHZRESULT JHZAPI JHDrawBitmapIst(
    HANDLE hdl,
    unsigned short index,
    short x,
    short y,
    unsigned short width,
    unsigned short height,
    unsigned long format,
    const JHBMP_SRC * pbmpsrc);
```

参数：

hdl	[IN] 实例句柄，与控制卡相对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
x	[IN] 图片显示位置左上角的 X 坐标。
y	[IN] 图片显示位置左上角的 Y 坐标。
width	[IN] 图片显示区域宽度。
height	[IN] 图片显示区域高度。
Format	[IN] 图片显示格式，如下所示：
	位 31-8: 保留, 为 0;
	位 7- 6: 水平对齐 HALIGN
	00: 左对齐;
	01: 水平居中对齐;
	10: 右对齐;
	位 5- 4:
	00: 上对齐;
	01: 垂直居中对齐;
	10: 下对齐;
	位 3-0: 保留, 为 0;
pbmpsrc	[IN] 指向存储图片源信息的内存空间，图片源信息结构体 BMP_SRC 如下：

```
struct tagBitmapSource{
    unsigned short type;
    unsigned short version;
    unsigned long size;
    unsigned char* pdata;
} JHBMP_SRC;
type: 图片类型, 始终为 0, 表示为文件图片;
version: 版本, 当前版本为 0;
size: 图片源数据大小;
pdata: 图片源数据; data 为动态数组, 大小由 size 决定;
```

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象
JR_NOT_ENOUGH_CORE	内存不足

功能描述:

在批处理文件中索引为index的位置上添加立即显示图片指令。显示的图片数据为pbmpsrc.pdata指向的内容, 若为文件图片则表示显示图片ID为pbmpsrc.pdata的图片。显示的区域为x, y, width, height确定的矩形内。

211. JHSpeechSynthesisStartIst (开始语音合成指令)

```
JHZRESULT JHZAPI JHSpeechSynthesisStartIst(
    HANDLE hdl,
    unsigned short index,
    unsigned long option,
    unsigned short num,
    const char *pformatstring);
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
option	[IN] 操作选项。 位 31-17: 保留; 位 17: 0 不重复, 1 自动重复; 位 16: 是否等待合成, 为 0 表示立即语音合成, 为 1 表示等待上一次语音合成结束后再合成本次语音; 位 15-8: 背景音乐, 0 表示无背景音乐, 1-15 表示背景音乐编号; 位 7-0: 字符编码; 0x00: GB2312 编码; 0x01: GBK 编码; 0x02: BIG5 编码; 0x03: UNICODE 编码;

其他保留：

num [IN] 字符个数, 1-1024 字节, (单位: 字节)。

Pformatstring [IN] 要语音合成的文本。

返回值:

JR_OK 成功

JR_INVALID_OBJECT 无效的对象

功能描述:

在批处理文件中索引为index的位置上添加开始语音合成指令。

212. JHSpeechSynthesisStopIst (停止语音合成指令)

JHZRESULT JHZAPI JHSpeechSynthesisStopIst(

HANDLE hdl,

unsigned short index);

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能描述:

在批处理文件中索引为index的位置上添加停止语音合成指令。

213. JHSpeechSynthesisPauseIst (暂停语音合成指令)

JHZRESULT JHZAPI JHSpeechSynthesisPauseIst(

HANDLE hdl,

unsigned short index);

参数:

hdl [IN] 实例句柄, 与控制卡相对应。

index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值:

JR_OK 成功

JR_INVALID_PARAMETER 参数错误

JR_INVALID_OBJECT 无效的对象

功能描述:

在批处理文件中索引为index的位置上添加暂停语音合成指令。

214. JHSpeechSynthesisRestartIst（恢复语音合成指令）

```
JHZRESULT JHZAPI JHSpeechSynthesisRestartIst(  
    HANDLE hdl,  
    unsigned short index);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在批处理文件中索引为index的位置上添加恢复语音合成指令。

215. JHSpeechSynthesisSleepIst（休眠语音模块指令）

```
JHZRESULT JHZAPI JHSpeechSynthesisSleepIst(  
    HANDLE hdl,  
    unsigned short index);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在批处理文件中索引为index的位置上添加休眠语音模块指令。

216. JHSpeechSynthesisWakeupIst（唤醒语音模块指令）

```
JHZRESULT JHZAPI JHSpeechSynthesisWakeupIst(  
    HANDLE hdl,  
    unsigned short index);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在批处理文件中索引为index的位置上添加唤醒语音模块指令。

217. JHSpeechSynthesisSetParaIst（设置语音参数指令）

```
JHZRESULT JHZAPI JHSpeechSynthesisSetPara (  
    HANDLE hdl  
    unsigned short index,  
    unsigned short para,  
    unsigned short rev0,  
    unsigned short rev1,  
    unsigned short rev2,  
);
```

参数：

hdl [IN] 实例句柄，与控制卡相对应。
index [IN] 指令在批处理文件中基于 1 开始的索引。
para [IN] 修改语音音量参数。
rev0 [IN] 保留。
Rev1 [IN] 保留。
Rev2 [IN] 保留。

返回值：

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述：

在批处理文件中索引为index的位置上添加设置语音参数指令。

218. JHSetIOStateIst（设置 IO 输出状态指令）

```
JHZRESULT JHZAPI JHSetIOStateIst (  
    HANDLE hdl  
    unsigned short index,  
    unsigned char ch,
```

```
        unsigned char state
    );
```

参数:

hdl	[IN] 实例句柄, 与控制卡相对应。
index	[IN] 指令在批处理文件中基于 1 开始的索引。
ch	[IN] IO 通道号, 有效值为 0-7, 不同的硬件可能会有差异。
state	[IN] 输出状态, 0 输出低电平 (0V), 1 输出高电平 (3.3V)。

返回值:

JR_OK	成功
JR_INVALID_PARAMETER	参数错误
JR_INVALID_OBJECT	无效的对象

功能描述:

在批处理文件中索引为index的位置上添加设置IO输出状态指令。

注意: IO 引脚最大输出电流约为 15mA, 连接到外接设备时, 要串联 330 欧姆电阻, 防止损坏字库卡。