Windows Base

HDACS API Reference Manual

Version 2.4.3

Table of Contents

Rev History

Version	Description
V2.0	First edition
V2.1	Add ECU-680 commands
V2.21	hacInitial modify
V2.3	Add 4.3.29 hacAddCardFingerPrintEx (Insert Finger pattern)
V2.4	Add RAC-970 Series & RAC-2000WS/WSN
V2.4.2	Add RAC-852 Series
V2.4.3	Add RAC-820PxF Series

This API reference manual is provided to help developers access software via API implementation therefore expediting their software development time. It uses Virtual Studio .NET Professional 2003 C++ as its software development tool.

Chapter 1 System Requirement

1. The DLL (Dynamic Link Library) supports Multi threading technology.
2. The DLL (Dynamic Link Library) support Delphi, VB6 (Visual Basic), VC(Visual C++), and C# development tools.
3. Four types sample code：C#2005.

Chapter 2 Hardware Supported List

RAC-2000 series	:	RAC-2000G, RAC-2000P, RAC-2000PS, RAC-2000PV, RAC-2000PN, RAC-2000PSN. RAC-2000WS, RAC-2000WSN. HDP-100, HDP-100S
HDE-100 series	:	HDE-100
RAC-960 series	:	RAC-960PE, RAC-960PM RAC-960PEF, RAC-960PMF
RAC-970 series	:	RAC-970PE, RAC-970PM RAC-970PEF, RAC-970PMF
RAC-940 series	:	RAC-940PE, RAC-940PM
RAC-2200 series	:	RAC-2200
RAC-2400 series	:	RAC-2400 series' SDK are different from other RAC devices. It may have a exclusive SDK. Kindly refer to Hdacs_rac2400*.pdf for more RAC-2400 series information.
ECU-680 series	:	ECU-680PM
RAC-852 series	:	RAC-852PEFV,RAC-852PMFV,RAC-852PHFV
RAC-820PxF series	:	RAC-820PMF/RAC-820PEF

Chapter 3 Return Value

Define	Return Value	Description
HF_RET_SUCCESS	0	True
HF_ERR_HANDLE_WAIT_TIMEOUT	1125	Overtime when operate multi-threading programs.
HF_ERR_HANDLE_RELEASE	1126	Error during released multi-threading.
HF_ERR_PARAMETER	1001	Error an sending a parameter. Or device returned an error code. Kindly refer to appendix. ReturnCode。
HF_ERR_SOCKET_ERROR	1002	Socket or communication port read/write error. An error occurred during asynchronous read/write.
HF_ERR_DATA_LENGTH	1003	Data length too short, device returned an invalid data length.
HF_ERR_RESPOND_LENGTH	1103	Length of packet small then reqeust
HF_ERR_HANDLE_INVALID	1004	Invalid control handler received. Invalid hComm value,
HF_ERR_RESPOND_ENDCHAR	1005	Error in the packet no. returned.
HF_ERR_RESPOND_CRC16	1006	Error of 16-bit Cyclic Redundancy Check (CRC-16) returned.
HF_ERR_SEND_CRC16	1106	Error of 16-bit Cyclic Redundancy Check (CRC-16) set.
HF_ERR_SEND_CMD	1007	PC sends wrong order to device or device does not support this function.
HF_ERR_SEND_RW	1008	An error occurred while performing read/write to slave device.
HF_ERR_SEND_OVERLENGTH	1009	Data length transmitted exceeded max. allowed length.
HF_ERR_RESPOND_NORECORDS	1010	No data was retrieved
HF_ERR_RESPOND_EXCEPT	4445	An error while reading device data or records
HF_ERR_WAIT_TIMEOUT	1025	Operation timed out during asynchronous read/write.
HF_ERR_WAIT_FAILED	1026	Operation error during asynchronous read/write.
HF_ERR_WAIT_NODATA	2225	Data was not retrieved during asynchronous read/write.

Chapter 4 RAC Series Functions

4.1 Communication – API (Level 0)

4.1.1 hacOpenChannelEX (Open Communication Channel)

Hardware	RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series	
API	hacOpenChannelEX	
Function	int __stdcall hacOpenChannelEX (char *sComm, unsigned int iPort, int iCheckStatus, HANDLE *hComm, unsigned int iTimeout)	
Purpose	Open TCP/IP or COM Port	
Arguments	sComm	For COM Port: sComm is COM1-COM128 For TCP/IP: sComm is IP address, ex.172.16.1.1
	iPort	For COM Port : iPort is Baudrate (1200/2400/4800/9600/19200/38400) All Hundure controllers baudrate should be 19200. For TCP/IP iPort is port number. Ex.: 4660
	iCheckStatus	Whenn connect with device through TCP/IP converter, like eP132 or BF430, Whenn receive value 2, which is meaning that “ Check if converter connect with internet success”. Whenn return value 1 which is meaning that no need to check interent situation.
	hComm:	For COM Port : Handle value if it returns true For TCP/IP: Socket value if it returns true.

	iTimeout	Using the parameter When connecting through TCP/IP, Overtime of connecting is base on millisecond.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application	However make order or retrieval data, you need call this function first before you operate other functions. Note: Whenn using TCP/IP converter, it is recommended NOT to stay online with the computer except during polling, perform connection only Whenn upload or download of data is required. Like BF-430 or eP-132, socket connection can only be performed in a short period of time (depending upon the parameter settings of the device), so it is advised to disconnect Whenn data transfer is not needs.If the connecting is unusual, please close the firewall and try again. In order to keep compatibility, hacOpenChannel parameter needs reserve.	

Remark	After open RS-232(COM 1) or TCP/IP(Port 4660) communication interface, call htaCloseChannel to close communication port.
	
	VC Code:
	
	int iReturn=0;
	char cIP[20];
	memset(cIP,0,sizeof(cIP));
	int iPort=0;
	int iPort=0;
	HANDLE ghComm=NULL;
	iPort=GetDlgItemInt(IDC_EDITPort,NULL);
	iReturn =hacOpenChannelEX(cIP,iPort,2,&ghComm,5000);
	if (iReturn!=0)
	{
	strcpy(sDisplay,"Open Channel: Error Number:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	else
	{	

```
        strcpy(sDisplay,"Open Channel: OK!,HANDLE:");
        sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)ghComm);
    }

    iReturn =hacCloseChannel(ghComm);
    if (iReturn!=0)
    {
        strcpy(sDisplay,"Close Channel: Error Number:");
        sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
    }
    else
    {
        strcpy(sDisplay,"Close Channel: OK!");
    }
```

4.1.2 hacCloseChannel (Close Communication Channel)

Hardware	RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series	
API	hacCloseChannel	
Function	int __stdcall hacCloseChannel (HANDLE hComm)	
Purpose	Close communication port	
Arguments	hComm	Handle value to be closed; if handle > 1000 then it is a TCP/IP port.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	Please refer to hacOpenChannelEX sample.	

4.2 Swipe card and Retrieve data – API (Level 1)

4.2.1 hacPolling (Polling Device)

Hardware	RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series	
API	hacPolling	
Function	int __stdcall hacPolling (int iNodeID, int iPrevRecord, stPollList *stRecord int *iRecord, HANDLE hComm, unsigned int iTimeout, int iCardType)	
Purpose	Sends polling command to device and wait for the returned data.	
Arguments	iNodeID	Device ID
	iPrevRecord	No. of records previously retrieved
	stRecord	Structure that points to the contents of data retrieved.
	iRecord	No. of records retrieved
	hComm	Com port or TCP/IP handle value
	iTimeout	Operationg timed out (millisencond)
	iCardType	Indicates Whether the card no. is compressed or not. Value is under 4 which is card no situation. Ex: 0 signifies uncompressed card number of RAC-2000G. 1 signifies compressed card number of RAC-2000G. 2 signifies card no. retrieval mode of RAC-2200. 3 signifies device of RAC-940/960/970series. Value is over 4 which is data retrieval mode. When the 4th of value is 1 (0x10), which is auto delivery mode. Note: Auto Deliver Mode only support RAC-2000PN and RAC-2000PSN. Ex: During auto deliver mode,

		0x00 and 0x01 signifies uncompressed card and compressed card number in normol mode. 0x10 and 0x11 signifies uncompressed card and compressed card number in auto deliver mode. Default is normol mode.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	

Application	{
	char cEventCode[5]; //Event Code
	When using RAC-940/960/970 series, the swiped card event has 4 digits. First and second digits signifies duty shift. Remaining digits signifies swiped status. Other devices are fixed front digits as “00”.
	char cDateTime[20]; //Date & Time
	char cCard[20]; //Card Number
	char cDeviceID[10]; //Device ID
	char cReaderID[10]; //Reader ID
	} stPollList;
	
	Note: RAC-2000P/PV/PS/PN/PSN/WS/WSN, HDP-100/100S support uncompressed card no only.
	RAC-2000G supports compressed card no. only.	

Remark	VC Code:
	
	BOOL Polling(int iDevice,int iCardType)
	{
	int iReturn=0;
	int iRecord=0;
	int iLoop=0;
	stPollList stRecord[256];
	if (giDeviceType==1)
	{
	iReturn =hac34Polling (iDevice,giPrevRecord[iDevice],stRecord,&iRecord,ghComm,100,iCardType);
	}
	else
	{
	iReturn =hacPolling	

	(iDevice,giPrevRecord[iDevice],stRecord,&iRecord,ghComm,100,iCard Type);
	}
	
	if (iReturn!=0)
	{
	return false;
	}
	giPrevRecord[iDevice]=iRecord;
	for(iLoop=0;iLoop<iRecord;iLoop++)
	{
	//Loop AddList
	AddListViewItems(hWndListView,&stRecord[iLoop]);
	}
	return true;
	}

4.3 Device – API (Leval 2)

4.3.1 hacGetDateTime (Reads the device's date and time)

Hardware	RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series	
API	hacGetDateTime	
Function	int __stdcall hacGetDateTime (int iNodeID, char *cDate, char *cTime, HANDLE hComm, unsigned int iTimeout)	
Purpose	Reads the date and time from the device	
Arguments	iNodeID	Device ID
	cDate	Pointer to date retrieved, format:yyyymmddw
	cTime	Pointer to time retrieved, format:hhmmss
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	
	int iReturn=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	char cDate[20];
	char cTime[20];
	memset(cDate,0,sizeof(cDate));
	memset(cTime,0,sizeof(cTime));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	iReturn =hacGetDateTime(iNodeID,cDate,cTime,ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Device Time:");	

	strcat(sDisplay,cDate);
	strcat(sDisplay," ");
	strcat(sDisplay,cTime);
	}
	else
	{
	strcpy(sDisplay,"Get DateTime: Error Number:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.2 hacSetDateTime (Sets the device's date and time)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series</td></tr>
<tr><td>API</td><td colspan="2">hacSetDateTime</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacSetDateTime (
int iNodeID,
char *cDate,
char *cTime,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Sets the date and time into the device</td></tr>
<tr><td rowspan="5">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cDate</td><td>Pointer to dateto be set, format:yyyymmddw</td></tr>
<tr><td>cTime</td><td>Pointer to time to be set, format:hhmmss</td></tr>
<tr><td>hComm</td><td>COM Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
char cDate[20];
char cTime[20];
memset(cDate,0,sizeof(cDate));
memset(cTime,0,sizeof(cTime));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
int iPos=0;
SYSTEMTIME st;
GetLocalTime(&st);
iPos=0;
sprintf(cDate+iPos,"%04d",st.wYear);
iPos=iPos+4;</td></tr>
</table>

```
sprintf(cDate+iPos,"%02d",st.wMonth);
iPos=iPos+2;
sprintf(cDate+iPos,"%02d",st.wDay);
iPos=iPos+2;
sprintf(cDate+iPos,"%01d",(st.wDayOfWeek));

iPos=0;
sprintf(cTime+iPos,"%02d",st.wHour );
iPos=iPos+2;
sprintf(cTime+iPos,"%02d",st.wMinute );
iPos=iPos+2;
sprintf(cTime+iPos,"%02d",st.wSecond );

iReturn =hacSetDateTime(iNodeID,cDate,cTime,ghComm,1000);
if (iReturn==0)
{
    strcpy(sDisplay,"Set DateTime:OK.");
}
else
{
    strcpy(sDisplay,"Set DateTime: Error Number:");
    sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
}
SetDlgItemText(IDC_EDITReturn,sDisplay);
```

4.3.3 hacGetVersion(Reads the device's version)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series,RAC-852 series</td></tr>
<tr><td>API</td><td colspan="2">hacGetVersion</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacGetVersion (
int iNodeID,
char *cData,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Reads the device's version</td></tr>
<tr><td rowspan="4">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cData</td><td>Pointer to the retrieved device's version/ROM File Date</td></tr>
<tr><td>hComm</td><td>COM Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2">First Section: Device's Model No.. Ex: RAC-2000P
Second Section: Major, Monor Version. Ex: V1.00
Third Section: Year, Month, Date. Ex: 2003/01/27</td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
char cReceiveBuff[128];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iLoop=0;
int iReadLen=0;
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
iReturn =hacGetVersion(iNodeID,cReceiveBuff,ghComm,1000);
if (iReturn==0)
{
strcpy(sDisplay,"Get Version:");
strcat(sDisplay,cReceiveBuff);
}</td></tr>
</table>

	else
	{
	strcpy(sDisplay,"Get Version:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.4 hacRelayAction(Set Relay Status)

Hardware	RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series	
API	hacRelayAction	
Function	int __stdcall hacRelayAction (int iNodeID, char cAction, char cMask, HANDLE hComm, unsigned int iTimeout)	
Purpose	Sets relay status which activates the relay	
Arguments	iNodeID	Device ID
	cAction	Relay's action(Door Relay ON/OFF, Alarm Relay ON/OFF) Byte:1 RAC-2000 series and RAC-2200 series Bit 0: Relay1 0=OFF 1=ON(Door1) Bit 1: Relay2 0=OFF 1=ON (Alarm1) Bit 2: Relay3 0=OFF 1=ON (Door2) Bit 3: Relay4 0=OFF 1=ON (Alarm2) Bit 4-7: Reserved RAC-940 series and RAC-960/970 series： Bit 0 : Open Door, Relay (0=OFF 1=ON) Bit 1 : Alarm , Relay (0=OFF 1=ON) Bit 2 : Ring , Relay (0=OFF 1=ON) Bit 3 : Door Bell relay (0=OFF 1=ON) Bit 4-7: Reserved

	cMask	Byte2: Relay to be activated Bit 0: Relay1 0=Deactivate. 1=Activate (Door1) Bit 1: Relay2 0= Deactivate, 1= Activate (Alarm1) Bit 2: Relay3 0= Deactivate, 1= Activate (Door2) Bit 3: Relay4 0= Deactivate, 1= Activate (Alarm2) Bit 4-7: Reserved RAC-940 series and RAC-960/970 series： Bit 0 : Open Door, Relay 0=Deactivate. 1=Activate Bit 1 : Alarm , Relay 0=Deactivate. 1=Activate Bit 2 : Ring , Relay 0=Deactivate. 1=Activate Bit 3 : Door Bell, Relay 0=Deactivate. 1=Activate Bit 4-7: Reserved
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	VC Code: iReturn =hacRelayAction(1,0x0f,0x0f,ghComm,1000);	

4.3.5 hacAddCard (Adds an Uncompessed Card)

Hardware	RAC-2000 series (Except RAC-2000G), RAC-940 series, RAC-960/970 series, RAC-2200 series	
API	hacAddCard	
Function	int __stdcall hacAddCard (int iNodeID, char *cCardNo, int iCardLen, char *cPassWord, int iPassLen, int iTimeZone, char cStatus, HANDLE hComm, unsigned int iTimeout)	
Purpose	Adds an uncompressed card	
Arguments	iNodeID	Device ID
	cCardNo	Pointer to ASCII code card no.
	iCardLen	Length of cCardNo
	iPassLen	Length of cPassword
	iTimeZone	Time Schedule
	cStatus	Status code like inspect on holidays, time schedule, blacklist…
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];	

	memset(cCardNumber,0,sizeof(cCardNumber));
	GetDlgItemText(IDC_EDITCard,cCardNumber,20);
	int iLoop=0;
	int iReadLen=0;
	iReturn
	=hacAddCard(iNodeID,cCardNumber,(int)strlen(cCardNumber),"
	",0x00,0x00,0x00,ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"(RAC2000P Support Only)Add Card:OK:");
	strcat(sDisplay,cCardNumber);
	}
	else
	{
	strcpy(sDisplay,"(RAC2000P Support Only)Add Card:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.6 hacDelCard(Delete an Uncompressed Card)

Hardware	RAC-2000 series (Except RAC-2000G), RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series, RAC-820PxF series	
API	hacDelCard	
Function	int __stdcall hacDelCard (int iNodeID, char *cCardNo, int iCardLen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Delete an uncompressed card	
Arguments	iNodeID	Device ID
	cCardNo.	Pointer to card no. to be deleted
	iCardLen	Length cCardNo
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];
	memset(cCardNumber,0,sizeof(cCardNumber));
	GetDlgItemText(IDC_EDITCard,cCardNumber,20);
	int iLoop=0;
	int iReadLen=0;
	iReturn
	=hacDelCard(iNodeID,cCardNumber,(int)strlen(cCardNumber),
	ghComm,1000);	

	if (iReturn==0)
	{
	strcpy(sDisplay,"(RAC2000P Support Only)Delete Card:OK:");
	strcat(sDisplay,cCardNumber);
	}
	else
	{
	strcpy(sDisplay,"(RAC2000P Support Only)Delete Card:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.7 hacGetSensor(Reads Sensor Status)

Hardware	RAC-2000 series, RAC-940 series, RAC-960/970 series, RAC-2200 series, RAC-852 series	
API	hacGetSensor	
Function	int __stdcall hacGetSensor (int iNodeID, int * iSensor, HANDLE hComm, unsigned int iTimeout)	
Purpose	Reads the status of the sensor	
Arguments	iNodeID	Device ID
	iSensor	Pointer to the returned status of sensor
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application	RAC-2000 series： -Bit 0 : Reads status of Push Button1, 1→Close, 0→Open -Bit 1 : Reads status of Push Button2, 1→Close, 0→Open -Bit 2 : Reads status of Door Sensor1, 1→Close, 0→Open -Bit 3 : Reads status of Door Sensor2, 1→Close, 0→Open -Bit 4 : Reads status of Case Sensor1, 1→Close, 0→Open -Bit 5 : Reads status of Case Sensor2, 1→Close, 0→Open -Bit 6 : Reads status of Case Sensor3, 1→Close, 0→Open -Bit 7 : Reads status of Case Sensor4, 1→Close, 0→Open RAC-2200 series： -Bit 0 : Reads status of IN1, 1→Close, 0→Open. (RAC-2200 fire activation) -Bit 1 : Reads status of IN2, 1→Close, 0→Open. (RAC-2200 alarm activation) -Bit 2 ~ Bit 7: Reserved RAC-940 series / RAC-960/970 series： Byte1: Sensor status -Bit 0 : Reads status of Case Sensor, 1→Close, 0→Open -Bit 1 : Reads status of Door Sensor, 1→Close, 0→Open	

	-Bit 2 : Reads status of Push Button, 1→Close, 0→Open -Bit 3~7 : Reserved Byte2: Relay status -Bit 0 : Relay of Open door, 1→Activate, 0→Inactivate -Bit 1 : Relay of Siren , 1→Activate, 0→Inactivate -Bit 2 : Relay of Alarm , 1→Activate, 0→Inactivate -Bit 3 : Relay of Door Bell, 1→Activate, 0→Inactivate -Bit 4~7 : Reserved

Remark	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iSensor=0x00;
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	iReturn =hacGetSensor(iNodeID,&iSensor,ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Read Sensor:");
	sprintf(sDisplay+strlen(sDisplay),"%02X ",(BYTE)iSensor);
	}
	else
	{
	strcpy(sDisplay,"Read Sensor:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.8 hacGetEEData (Reads the Device's EEPROM Value)

Hardware	RAC-2000 series, RAC-2200 series	
API	hacGetEEData	
Function	int __stdcall hacGetEEData (int iNodeID, char *cEEData, int *iReceiveDataLen, unsigned int *iEEAddr, int *iEELen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Reads the EEPROM value from the device	
Arguments	iNodeID	Device ID
	cEEData	Pointer to contents of EEPROM retrieved.
	iReceiveDataLen	Pointer to length of EEPROM retrieved.
	iEEAddr	Retrieved EEPROM location
	iEELen	Expected EEPROM length, max. 240 bytes
	hComm	Com Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	iReturn =hacGetEEData(iNodeID,cReceiveBuff,
	&iReadLen,0x0069,0x08,ghComm,1000);
	if (iReturn==0)
	{	

```
            strcpy(sDisplay,"Read EEPROM:");
            for(iLoop=0;iLoop<iReadLen;iLoop++)
            {
                sprintf(sDisplay+strlen(sDisplay),"%02X
",(BYTE)cReceiveBuff[iLoop]);
            }
        }
        else
        {
            strcpy(sDisplay,"Read EEPROM:Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

4.3.9 hacSetEEData (Sets EEPROM Value)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-2000 series, RAC-2200 series</td></tr>
<tr><td>API</td><td colspan="2">hacSetEEData</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacSetEEData (
int iNodeID,
char *cEEData,
unsigned int iEEAddr,
int iEELen,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Writes data into the device’s EEPROM</td></tr>
<tr><td rowspan="6">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cEEData</td><td>Pointer to contents of EEPROM to be set. Content contains storage location of data.</td></tr>
<tr><td>iEEAddr</td><td>EEPROM location to be set</td></tr>
<tr><td>iEELen</td><td>EEPROM length to set, max. 240 bytes</td></tr>
<tr><td>hComm</td><td>Com Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
cReceiveBuff[0]=0x05;
iReturn =hacSetEEData(iNodeID,(unsigned char*)cReceiveBuff,0x0069,0x01,ghComm,1000);
if (iReturn==0)
{
strcpy(sDisplay,”Write EEPROM:OK!”);</td></tr>
</table>

	}
	else
	{
	strcpy(sDisplay,”Write EEPROM:Error:”);
	sprintf(sDisplay+strlen(sDisplay),”%04X “,(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.10 hacAddZCard (Adds a Compressed Card)

Hardware	RAC-2000G	
API	hacAddZCard	
Function	int __stdcall hacAddZCard (int iNodeID, char *cCardNo, int iCardLen, char *cPassword, int iPassLen, char cStatus, HANDLE hComm, unsigned int iTimeout)	
Purpose	Adds a compressed card. All numbers should be digits.	
Arguments	iNodeID	Device ID
	cCardNo.	Pointer to ASCII code card number
	iCardLen	Length of cCardNo
	cPassword	Pointer to ASCII code password
	iPassLen	Length of cPassword. Sets to 0 if password is not provided.
	cStatus	Status Code like inspection on holidays, time schedule, blacklist…
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];
	memset(cCardNumber,0,sizeof(cCardNumber));	

```
    GetDlgItemText(IDC_EDITCard,cCardNumber,20);
    int iLoop=0;
    int iReadLen=0;

    iReturn
=hacAddZCard(iNodeID,cCardNumber,(int)strlen(cCardNumber),"
",0x00,0x00,ghComm,1000);
    if (iReturn==0)
    {
        strcpy(sDisplay,"(RAC2000G Support Only)Add Compress
Card:OK:");
        strcat(sDisplay,cCardNumber);
    }
    else
    {
        strcpy(sDisplay,"(RAC2000G Support Only)Add Compress
Card:Error!");
        sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
    }
    SetDlgItemText(IDC_EDITReturn,sDisplay);
```

4.3.11 hacDelZCard(Delete a Compressed Card)

Hardware	RAC-2000G	
API	hacDelZCard	
Function	int __stdcall hacDelZCard (int iNodeID, char *cCardNo, int iCardLen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Delete a compressed card.	
Arguments	iNodeID	Device ID
	cCardNo	Pointer to ASCII code card number
	iCardLen	Length of cCardNo
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];
	memset(cCardNumber,0,sizeof(cCardNumber));
	GetDlgItemText(IDC_EDITCard,cCardNumber,20);
	int iLoop=0;
	int iReadLen=0;
	iReturn
	=hacDelZCard(iNodeID,cCardNumber,(int)strlen(cCardNumber),
	ghComm,1000);
	if (iReturn==0)
	{	

	strcpy(sDisplay,"(RAC2000G Support Only)Delete Compress Card:OK:");
	strcat(sDisplay,cCardNumber);
	}
	else
	{
	strcpy(sDisplay,"(RAC2000G Support Only)Delete Compress Card:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.12 hacGetRAMData (Reads the Data within the Device's RAM)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-2000 series, RAC-960/970 series, RAC-2200 series,
RAC-820PxF series, RAC-820PxF series</td></tr>
<tr><td>API</td><td colspan="2">hacGetRAMData</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacGetRAMData (
int iNodeID,
char *cRAMData,
int *iReceiveDataLen，
unsigned int iRAMAddr,
int iRAMLen,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Reads the data from the device's RAM</td></tr>
<tr><td rowspan="8">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cRAMData</td><td>Pointer to the data retrieved from RAM</td></tr>
<tr><td>iReceiveDataLen</td><td>Pointer to the retruned RAM's length</td></tr>
<tr><td>iRAMAddr</td><td>Returned RAM's address</td></tr>
<tr><td>iRAMLen</td><td>Expected RAM's length, max. of 240 bytes</td></tr>
<tr><td>hComm</td><td>COM Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
int iLoop=0;
int iReadLen=0;
iReturn =hacGetRAMData(iNodeID,cReceiveBuff,
&iReadLen,0x00,0x40,ghComm,1000);</td></tr>
</table>

	if (iReturn==0)
	{
	strcpy(sDisplay,"Read RAM:");
	for(iLoop=0;iLoop<iReadLen;iLoop++)
	{
	sprintf(sDisplay+strlen(sDisplay),"%02X
	",(BYTE)cReceiveBuff[iLoop]);
	}
	}
	else
	{
	strcpy(sDisplay,"Read RAM:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.13 hacSetRAMData (Sets RAM Value)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-2000 series, RAC-960/970 series, RAC-2200 series,
RAC-820PxF series</td></tr>
<tr><td>API</td><td colspan="2">hacSetRAMData</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacSetRAMData (
int iNodeID,
char *cRAMData,
unsigned int iRAMAddr,
int iRAMLen,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Writes data into the device’s RAM</td></tr>
<tr><td rowspan="6">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cRAMData</td><td>Pointer to contents of RMA to be set. Contents contain storage location of data.</td></tr>
<tr><td>iRAMAddr</td><td>RAM location to be set</td></tr>
<tr><td>iRAMLen</td><td>RAM length to be set, max. of 240 bytes</td></tr>
<tr><td>hComm</td><td>COM Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
cReceiveBuff[0]=0x01;
cReceiveBuff[1]=0x01;
iReturn =hacSetRAMData(iNodeID,(unsigned char*)cReceiveBuff,0x000020,0x02,ghComm,1000);
if (iReturn==0)</td></tr>
</table>

	{
	strcpy(sDisplay,"Write RAM:OK!");
	}
	else
	{
	strcpy(sDisplay,"Write RAM:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.14 hacAddVisitorCard(Adds a Visitor Card)

Hardware	RAC-2000 series, RAC-2200 series, RAC-820PxF series	
API	hacAddVisitorCard	
Function	int __stdcall hacAddVisitorCard (int iNodeID, char *cCardNo, int iCardStart, int iCardLen, char *cStartDate, char *cStartTime, char * cEndDate, char * cEndTime, int iWeek, int iTimes, int iSerial,, int iAnti, HANDLE hComm, unsigned int iTimeout)	
Purpose	Adds an uncompressed visitor card.	
Arguments	iNodeID	Device ID
	cCardNo	Pointer to ASCII code card no
	iCardStart	Index code of card no
	iCardLen	Length of card no
	cStartDate	Pointer to card's date of validity, format:yyyymmdd
	cStartTime	Pointer to card's time of validity, format:hhmmss
	cEndDate	Pointer to card's date of expiry, format:yyyymmdd
	cEndTime	Pointer to card's time of expiry, format:hhmmss
	iWeek	Starts from higher byte to lower byte. Monday to Sunday, input 0 at the lower byte. Bit6: Mon, Bit5: Tue, Bit4: Wed, Bit3: Thu, Bit2: Fri, Bit1: Sat, Bit0: Sun
	iTimes	Vaild Frequency of card swipe
	iSerial	Location of card no to be added. There are a total of 50 cards can be added from 0-49.
	iAnti	Anti-passback situation. Please refer to appendix.
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)

Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.
Application	

Remark	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];
	memset(cCardNumber,0,sizeof(cCardNumber));
	GetDlgItemText(IDC_EDITCard,cCardNumber,20);
	int iLoop=0;
	int iReadLen=0;
	iReturn =hacAddVisitorCard(iNodeID,cCardNumber,0x01,
	(int)strlen(cCardNumber),
	"20010101","000000","20151231","235959",
	0xff,2000,0x00,0x01,ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Add Visitor Card:OK:");
	strcat(sDisplay,cCardNumber);
	}
	else
	{
	strcpy(sDisplay,"Add Visitor Card:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.15 hacDelVisitorCard(Deletes a Visitor Card)

Hardware	RAC-2000 series, RAC-2200 series, RAC-820PxF series	
API	hacDelVisitorCard	
Function	int __stdcall hacDelVisitorCard (int iNodeID, int iSerial,, HANDLE hComm, unsigned int iTimeout)	
Purpose	Deletes an uncompressed visitor card.	
Arguments	iNodeID	Device ID
	iSerial	Location of the card no to be deleted. There are a total of 50 cards can be deleted from 0-49
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	RAC-2000 series provides 50 sets guest card. Please refer to appendix 1.8, 1.12 and 1.13.
	
	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];
	memset(cCardNumber,0,sizeof(cCardNumber));
	GetDlgItemText(IDC_EDITCard,cCardNumber,20);
	int iLoop=0;
	int iReadLen=0;
	iReturn =hacDelVisitorCard(iNodeID,0x01,ghComm,1000);
	if (iReturn==0)	

	{
	strcpy(sDisplay,"Delete Visitor Card:OK:");
	strcat(sDisplay,cCardNumber);
	}
	else
	{
	strcpy(sDisplay,"Delete Visitor Card:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.16 hacGetFlashData (Reads Device's Flash Value)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series	
API	hacGetFlashData	
Function	int __stdcall hacGetFlashData (int iNodeID, char *cFlashData, int *iReceiveDataLen, unsigned int *iFlashAddr, int *iFlashLen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Reads the Flash value from the device.	
Arguments	iNodeID	Device ID
	cFlashData	Pointer to contents of Flash retrieved.
	iReceiveDataLen	Pointer to length of Flash retrieved.
	iFlashAddr	Retrieved Flash location
	iFlashLen	Expected Flash length, max. 240 bytes
	hComm	Com Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	iReturn	

	=hacGetFlashData(iNodeID,cReceiveBuff,&iReadLen,0x0069,0x08,
	ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Read Flash:");
	for(iLoop=0;iLoop<iReadLen;iLoop++)
	{
	sprintf(sDisplay+strlen(sDisplay),"%02X
	",(BYTE)cReceiveBuff[iLoop]);
	}
	}
	else
	{
	strcpy(sDisplay,"Read Flash:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.17 hacSetFlashData(Sets Flash Value)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-940 series, RAC-960/970 series, RAC-852 series</td></tr>
<tr><td>API</td><td colspan="2">hacSetFlashData</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacSetFlashData (
int iNodeID,
char *cFlashData,
unsigned int iFlashAddr,
int iFlashLen,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Writes data into the device’s Flash</td></tr>
<tr><td rowspan="6">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cFlashData</td><td>Pointer to contents of Flash to be set. Content contains storage location of data.</td></tr>
<tr><td>iFlashAddr</td><td>Flash location to be set</td></tr>
<tr><td>iFlashLen</td><td>Flash length to set, max. 240 bytes</td></tr>
<tr><td>hComm</td><td>Com Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
cReceiveBuff[0]=0x05;
iReturn =hacSetFlashData(iNodeID,(unsigned char*)cReceiveBuff,0x0069,0x01,ghComm,1000);
if (iReturn==0)
{</td></tr>
</table>

	strcpy(sDisplay,"Write Flash:OK!");
	}
	else
	{
	strcpy(sDisplay,"Write Flash:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.18 hacGetParaData (Reads General Parameter)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series</td></tr>
<tr><td>API</td><td colspan="2">hacGetParaData</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacGetParaData (
int iNodeID,
char *cFlashData,
int *iReceiveDataLen,
HANDLE hComm,
unsigned int iTimeout</td></tr>
<tr><td>Purpose</td><td colspan="2">Reads the parameter value from the device’s Flash.</td></tr>
<tr><td rowspan="5">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cFlashData</td><td>Pointer to contents of Flash retrieved.</td></tr>
<tr><td>iReceiveDataLen</td><td>Pointer to length of Flash retrieved.</td></tr>
<tr><td>hComm</td><td>COM Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:
int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
int iLoop=0;
int iReadLen=0;
iReturn
=hacGetFlashData(iNodeID,cReceiveBuff,&iReadLen,0x0069,0x08,ghComm,1000);
if (iReturn==0)
{
strcpy(sDisplay,"Read Flash:");
for(iLoop=0;iLoop<iReadLen;iLoop++)</td></tr>
</table>

```
            {
                sprintf(sDisplay+strlen(sDisplay),"%02X
",(BYTE)cReceiveBuff[iLoop]);
            }
        }
        else
        {
            strcpy(sDisplay,"Read Flash:Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

4.3.19 hacSetParaData (Sets General Parameter)

<table>
<tr><td>Hardware</td><td colspan="2">RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series</td></tr>
<tr><td>API</td><td colspan="2">hacSetParaData</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hacSetParaData (
int iNodeID,
char *cFlashData,
int iFlashLen,
HANDLE hComm,
unsigned int iTimeout
)</td></tr>
<tr><td>Purpose</td><td colspan="2">Writes data into the device’s Flash</td></tr>
<tr><td rowspan="5">Arguments</td><td>iNodeID</td><td>Device ID</td></tr>
<tr><td>cFlashData</td><td>Pointer to contents of Flash to be set.
Content contains storage location of data.</td></tr>
<tr><td>iFlashLen</td><td>Flash length to set, max. 240 bytes</td></tr>
<tr><td>hComm</td><td>COM Port or TCP/IP handle value</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond)</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
cReceiveBuff[0]=0x05;
iReturn =hacSetFlashData(iNodeID,(unsigned char*)cReceiveBuff,0x0069,0x01,ghComm,1000);
if (iReturn==0)
{
strcpy(sDisplay,"Write Flash:OK!");</td></tr>
</table>

	}
	else
	{
	strcpy(sDisplay,"Write Flash:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.20 hacGetSysParaData (Reads System Parameter)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series	
API	hacGetSysParaData	
Function	int __stdcall hacGetSysParaData (int iNodeID, char *cFlashData, int *iReceiveDataLen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Reads the system parameter value from the device.	
Arguments	iNodeID	Device ID
	cFlashData	Pointer to contents of Flash retrieved.
	iReceiveDataLen	Pointer to length of Flash retrieved.
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));	

```
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	iReturn
=hacGetFlashData(iNodeID,cReceiveBuff,&iReadLen,0x0069,0x08,ghC
omm,1000);
	if (iReturn==0)
	{
		strcpy(sDisplay,"Read Flash:");
		for(iLoop=0;iLoop<iReadLen;iLoop++)
		{
			sprintf(sDisplay+strlen(sDisplay),"%02X
",(BYTE)cReceiveBuff[iLoop]);
		}
	}
	else
	{
		strcpy(sDisplay,"Read Flash:Error:");
		sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);
```

4.3.21 hacSetSysParaData (Sets System Parameter)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series	
API	hacSetSysParaData	
Function	int __stdcall hacSetSysParaData (int iNodeID, char *cFlashData, int iFlashLen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Writes system parameter into the device's flash	
Arguments	iNodeID	Device ID
	cFlashData	Pointer to contents of Flash to be set. Content contains storage location of data.
	iFlashLen	Flash length to set, max. 240 bytes
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code：
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	cReceiveBuff[0]=0x05;
	iReturn =hacSetFlashData(iNodeID,(unsigned char*)cReceiveBuff,0x0069,0x01,ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Write Flash：OK!");
	}	

	else
	{
	strcpy(sDisplay,"Write Flash：Error：");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.22 hacGetMifare (Reads Mifare module parameter)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series	
API	hacGetMifare	
Function	int __stdcall hacGetMifare (int iNodeID, int iKeyType, int *iBlock, int *iStartDigit, int *iDigitLength, int *iCompact, HANDLE hComm, unsigned int iTimeout)	
Purpose	Reads the parameter value from MIFARE module of device.	
Arguments	iNodeID	Device ID
	iKeyType	0 signifies read CSN. 1 signifies read Key A, 2 signifies read Key B.
	IBlock	When iKeyType >0, point out read which block. (0x00~0x3F)
	iStartDigit	Read (0~15) data from which byte of current blcok
	iDigitLength	Pointer to lenth of digits retrieved (0~12)
	iCompact	The card no. is compressed or not. 0 signifies uncompressed. 1 signifies compressed.
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code：
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	iReturn =hacGetFlashData(iNodeID,cReceiveBuff,&iReadLen,0x0069,0x08, ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Read Flash：");
	for(iLoop=0;iLoop<iReadLen;iLoop++)
	{
	sprintf(sDisplay+strlen(sDisplay),"%02X ",(BYTE)cReceiveBuff[iLoop]);
	}
	}
	else
	{
	strcpy(sDisplay,"Read Flash：Error：");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.23 hacSetMifare (Sets Mifare Module Parameter)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series	
API	hacSetMifare	
Function	int __stdcall hacSetMifare (int iNodeID, int iKeyType, int iBlock, int iStartDigit, int iDigitLength, int iCompact, unsigned char *cKeyValue, HANDLE hComm, unsigned int iTimeout)	
Purpose	Writes the parameter value into MIFARE module of device.	
Arguments	iNodeID	Device ID
	iKeyType	0 signifies read CSN. 1 signifies read Key A, 2 signifies read Key B.
	iBlock:	When iKeyType >0, point out read which block.
	iStartDigit	Read (0~15) data from which byte of current blcok
	iDigitLength	Pointer to lenth of digits retrieved (0~12)
	iCompact	The card no. is compressed or not. 0 signifies uncompressed. 1 signifies compressed.
	cKeyValue	Flash length to set, max. 240 bytes
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code：
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];	

```
        memset(sDisplay,0,sizeof(sDisplay));
        unsigned char cReceiveBuff[280];
        memset(cReceiveBuff,0,sizeof(cReceiveBuff));
        int iNodeID=0x01;
        iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
        cReceiveBuff[0]=0x05;
        iReturn =hacSetFlashData(iNodeID,(unsigned
char*)cReceiveBuff,0x0069,0x01,ghComm,1000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Write Flash：OK!");
        }
        else
        {
            strcpy(sDisplay,"Write Flash：Error：");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

4.3.24 hacAddCardEx (Adds a card & display name)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series	
API	hacAddCardEx	
Function	int __stdcall hacAddCardEX(int iNodeID, char *cCardNo, int iCardLen, char *cPassWord, int iPassLen, char *cName, int iNameLen, int iTimeZone, char cStatus, HANDLE hComm, unsigned int iTimeout)	
Purpose	Adds an uncompressed card and display name. (Make sure the system parameter has been set display card number and name, otherwise will receive error code from device.)	
Arguments	iNodeID	Device ID
	cCardNo	Pointer to ASCII code card no.
	iCardLen	Length of Card No
	cPassWord	Pointer to ASCII code password
	iPassLen	Length of Password
	cNameLen	Pointer to ASCII code name. Only RAC-940/960/970 series can display name.
	iNameLen	Length of name
	iTimeZone	Time Schedule
	cStatus	Status code like inspect on holidays, time schedule, blacklist.
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		

Remark	VC Code：
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	char cCardNumber[20];
	memset(cCardNumber,0,sizeof(cCardNumber));
	GetDlgItemText(IDC_EDITCard,cCardNumber,20);
	char cName[20];
	memset(cName,0,sizeof(cName));
	GetDlgItemText(IDC_EDITCard4,cName,20);
	int iLoop=0;
	int iReadLen=0;
	
	iReturn
	=hacAddCardEX(iNodeID,cCardNumber,(int)strlen(cCardNumber),
	"",0x00,cName,(int)strlen(cName),0x00,0x00,ghComm,1000);
	
	if (iReturn==0)
	{
	strcpy(sDisplay,"(RAC940/960/970 Support Only)Add
	CardEX:OK:");
	strcat(sDisplay,cCardNumber);
	}
	else
	{
	strcpy(sDisplay,"(RAC940/960/970 Support Only)Add
	CardEX:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.3.25 hacInitial(Hardware Initializing)

Hardware	RAC-940 series, RAC-960/970 series, RAC-852 series, RAC-820PxF series	
API	hacInitial	
Function	int __stdcall hacInitial (int iNodeID, int iDeviceType, int iClearFlag, HANDLE hComm, unsigned int iTimeout)	
Purpose	Pointer to hardware data to initializing	
Arguments	iNodeID	Device ID
	iDeviceType	Parameter 3 signifies RAC-960/970 series. Parameter 4 signifies RAC-940 series.
	iClearFlag	What hardware data do you want to initial. Bit 0: 1 Delete all card numbers. Bit 1: 1 Initial all the tables Bit 2: 1 Delete swiped card and event records Bit 3: 1 General parameter initiation. Bit 4: 1 Only clear blacklist. Bit 5~7: Reserved
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	VC Code: iReturn =hacInitial(1,0x03,0xff,ghComm,1000);	

4.3.26 hacFingnerPrinterQueryMasterFP(Query Master Fingerprint)

Hardware	RAC-960/970PEF, RAC-960/970PMF, RAC-852 series	
API	hacFingerPrinterQueryMasterFP	
Function	int __stdcall hacFingerPrinterQueryMasterFP(int iNodeID,HANDLE hComm, unsigned char *cFingerPrinterData1,unsigned char *cFingerPrinterData2,int *iReturnCode,unsigned int iTimeOut)	
Purpose	Query Master fingerprint	
Arguments	hComm	COM Port or TCP/IP handle value
	iNodeID	Device ID
	CardLen	Length of card number
	cCardNo	Card number
	cFingerPrinterData1	First finger pattern retrieved
	cFingerPrinterData1	Second finger pattern retrieved
	iReturnCode	Error code returned by the device. Kindly refer to Appendix for description.
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

4.3.27 hacFingnerPrinterUpdateMasterFPV(Change Master Finger pattern)

Hardware	RAC-960/970PEF, RAC-960/970PMF, RAC-852 series	
API	hacFingerPrinterUpdateMasterFPV	
Function	int __stdcall hacFingerPrinterUpdateMasterFPV(int iNodeID,HANDLE hComm, unsigned char *cFingerPrinterData1,unsigned char *cFingerPrinterData2,int *iReturnCode,unsigned int iTimeOut)	
Purpose	Change Master finger pattern	
Arguments	hComm	COM Port or TCP/IP handle value
	iNodeID	Device ID
	cFingerPrinterData1	First finger pattern retrieved
	cFingerPrinterData1	Second finger pattern retrieved
	iReturnCode	Error code returned by the device. Kindly refer to Appendix for description.
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

4.3.28 hacFingnerPrinterDeleteFP(Delete Master Finger Pattern)

Hardware	RAC-960/970PEF, RAC-960/970PMF, RAC-852 series	
API	hacFingerPrinterDeleteFP	
Function	int __stdcall hacFingerPrinterDeleteFP(int iNodeID,HANDLE hComm, int *iReturnCode,unsigned int iTimeOut)	
Purpose	Delete Master finger pattern	
Arguments	hComm	COM Port or TCP/IP handle value
	iNodeID	Device ID
	iReturnCode	Error code returned by the device. Kindly refer to Appendix for description.
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

4.3.29 hacAddCardFingerPrintEx (Insert Finger Pattern)

Hardware	RAC-960/970PXF series, RAC-852PXFV series, RAC-820PxF series	
API	hacAddCardFingerPrintEx	
Function	Int __stdcall hacAddCardFingerPrintEx (int iNodeID,char *cCardNo,int iCardLen,char *cPassword,int iPassLen,int iTimeZone,char cStatus,unsigned char *cFingerPrinterData1,unsigned char *cFingerPrinterData2,HANDLE hComm, unsigned int iTimeOut)	
Purpose	Insert finger pattern into machine	
Arguments	iNodeID	Device ID
	cCardNo	Pointer to ASCII code card number
	iCardLen	Length of Card No
	cPassWord	Pointer to ASCII code password
	iPassLen	Length of Password
	iTimeZone	Time Schedule
	cStatus	Status code like inspect on holidays, time schedule, blacklist.
	cFingerPrinterData1	First finger pattern
	cFingerPrinterData2	Second finger pattern
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

4.3.30 hacFingerPrintQueryUser (Retrieve Finger Pattern)

Hardware	RAC-960/970PXF series, , RAC-852 series, RAC-820PxF series
API	hacFingerPrintQueryUser
Function	Int __stdcall hacFingerPrintQueryUser (int iNodeID, HANDLE hComm, int iCardLen, char *cCardNo ,unsigned char *cFingerPrinterData1,unsigned char *cFingerPrinterData2,int *iCardFormatLen, int *iReturnCode, unsigned int iTimeOut)
Purpose	Retrieve finger pattern

Arguments	iNodeID	Device ID
	iCardLen	Length of Card No
	cCardNo	Pointer to ASCII code card number
	cFingerPrinterData1	First finger pattern
	cFingerPrinterData2	Second finger pattern
	iCardFormatLen	Length of finger pattern
	iRetrunCode	Error code returned
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

4.4 Parameters API (Level 3)

4.4.1 hacSetLanMode (Set Active Polling Mode)

Hardware	RAC-2000PN/PSN	
API	hacSetLanMode	
Function	int __stdcall hacSetLanMode (int iNodeID, char cMode, HANDLE hComm, unsigned int iTimeout)	
Purpose	Set active polling mode of RAC-2000PN/PSN	
Arguments	iNodeID	Device ID
	cMode	0 is disable active polling mode. 1 is enable active polling mode
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	cReceiveBuff[0x0]=0x01;
	iReturn=hacSetLanMode(iNodeID,1,ghComm,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Enabled Active Send:OK!");
	}	

	else
	{
	strcpy(sDisplay,"Enabled Active Send:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

4.4.2 hacSetReader (Set Reader Parameter of RAC-2200)

Hardware	RAC-2200 series	
API	hacSetReader	
Function	int __stdcall hacSetReader (int iNodeID, int iReaderID, int iIndex, unsigned char *cSendData, int iSendDataLen, HANDLE hComm, unsigned int iTimeout)	
Purpose	Sets reader parameter value to RAC-2200.	
Arguments	iNodeID	Device ID
	iReaderID	Reader ID from 0-3 are for outside readers, ID 16-19 are for inside readers. kindly refer to appendix.
	iIndex	Read/Write index of reader
	cSendData	Parameter to set to reader
	iSendDataLen	Length of send data
	hComm	COM Port or TCP/IP handle value
	iTimeout	Operation timed out (millisecond)
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application	Set the reader parameter value of RAC-2200	
Remark	This function only support RAC-2200	

Sample	VC Code：
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];	

```
    memset(sDisplay,0,sizeof(sDisplay));
    unsigned char cReceiveBuff[280];
    memset(cReceiveBuff,0,sizeof(cReceiveBuff));
    int iNodeID=0x01;
    iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
    //control the beep sound of reader
    cReceiveBuff[0x0]=0x00;
    cReceiveBuff[0x1]=0x00;
    cReceiveBuff[0x2]=0x0D;
    cReceiveBuff[0x3]=0x0F;
    cReceiveBuff[0x4]=0x01;
    cReceiveBuff[0x5]=0x02;
    cReceiveBuff[0x6]=0x00;
    iReturn=hacSetReader(iNodeID,1,23,cReceiveBuff,7,ghComm,10
00);
    if (iReturn==0)
    {
        strcpy(sDisplay,"Set Reader:OK!");
    }
    else
    {
        strcpy(sDisplay,"Set Reader:Error:");
        sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
    }
    SetDlgItemText(IDC_EDITReturn,sDisplay);
```

Chapter 5 HDE-100 series Functions

5.1 Communication API

5.1.1 hsOpenChannel(Opens communication channel)

Hardware	HDE-100 series	
API	hsOpenChannel	
Function	int __stdcall hsOpenChannel(HANDLE *hComm, char *sComm, unsigned int iPort)	
Purpose	Opens TCP/IP or COM Port	
Arguments	hComm	The handle value if it returns true
	sComm	COM1-COM128 or IP address
	iPort	Baudrate (1200/2400/4800/9600/19200/38400) or TCP/IP port number
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	For TCP/IP: sComm: IP address, ex: 172.16.1.1 iPort: Port No., ex: 4660 hComm: Socket value if it returns true. Note: The returned TCP/IP handle value is added with 1000 to prevent having the same handle value with COM Port.	

Sample	VC Code:
	
	int iReturn=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	char cIP[20];
	memset(cIP,0,sizeof(cIP));
	int iPort=0;
	
	GetDlgItemText(IDC_EDITIP,cIP,20);	

	iPort=GetDlgItemInt(IDC_EDITPort,NULL);
	iReturn =hsELOpenChannel(&ghComm,cIP,iPort);
	if (iReturn!=0)
	{
	strcpy(sDisplay,"Open Channel: Error Number:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	else
	{
	strcpy(sDisplay,"Open Channel: OK!,HANDLE:");
	sprintf(sDisplay+strlen(sDisplay),"%04X
	",(WORD)ghComm);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

5.1.2 hsCloseChannel (Closes communication channel)

Hardware	HDE-100 series	
API	hsCloseChannel	
Function	int __stdcall hsCloseChannel(HANDLE hComm);	
Purpose	Closes communication port	
Arguments	hComm	COM Port or TCP/IP handle value
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application	If hComm value>1000, then it is an RS-232 interface.	
Remark		

Sample	VC Code:
	
	int iReturn=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	
	iReturn =hsELCloseChannel(ghComm);
	if (iReturn!=0)
	{
	strcpy(sDisplay,"Close Channel: Error Number:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	else	

	{
	strcpy(sDisplay,"Close Channel: OK!");
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

5.1.3 hsELWriteTable(Sends Table to HDE-100 series)

Hardware	HDE-100 series	
API	hsELWriteTable	
Function	int __stdcall hsELWriteTable(HANDLE hComm, int iELID, unsigned char *cTableData, int iTableLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Sends a single table to HDE-100 series, kindly refer to appendix.	
Arguments	hComm	COM Port or TCP/IP handle value
	iELID	Device ID
	cTableData	Pointer to the contents of table to be transmitted.
	iTableLen	Length of table to be transmitted.
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). Recommended time is 10ms and above.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	Table	Description
	Holiday	Location: 0-600
	Weekday	Location: 601-2378
	Time Zone	Location: 2379-6474
	A max. of 240 bytes data can be set.	

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];	

```
	//unsigned char cReceiveBuff[301];
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iWrittenLen=240;
	memset(cReceiveBuff,0xff,sizeof(cReceiveBuff));

	cReceiveBuff[0]=0x00;
	cReceiveBuff[1]=0x00;
	cReceiveBuff[2]=0xF0;
	cReceiveBuff[3]=0x00;

	cReceiveBuff[4]=0X07 ;
	cReceiveBuff[5]=0X31 ;
	cReceiveBuff[6]=0X31 ;
	cReceiveBuff[7]=0X31 ;
	cReceiveBuff[8]=0X32 ;
	cReceiveBuff[9]=0X40 ;
	cReceiveBuff[10]=0X31 ;

	iReturn=hsELWriteTable(ghComm,iNodeID,
cReceiveBuff,iWrittenLen,&iReturnCode,5000);

	if (iReturn==0)
	{
		strcpy(sDisplay,"Write Table:OK!");
	}
```

5.1.4 hsELReadTable(Reads HDE-100 series Table)

Hardware	HDE-100 series	
API	hsELReadTable	
Function	int __stdcall hsELReadTable(HANDLE hComm, int iELID, unsigned char *cTableData, int *iTableLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Reads a table fro HDE-100 series, kindly refer to appendix.	
Arguments	hComm	COM Port or TCP/IP handle value
	iELID	Device ID
	cTableData	The contents of data retrieved.
	iTableLen	The length of data retrieved.
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). Recommended time is 10ms and above.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	Table	Description
	Holiday	Location: 0-600
	Weekday	Location: 601-2378
	Time Zone	Location: 2379-6474
	A max. of 240 bytes data can be set.	

Sample	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;	

```
        int iReadLen=0;
        int iTotal=0;
        iTotal=2 ;
        memset(cReceiveBuff,0x00,sizeof(cReceiveBuff));

        cReceiveBuff[0]=0x59 ;
        cReceiveBuff[1]=0x02 ;

        cReceiveBuff[2]=0x60 ;
        cReceiveBuff[3]=0x02 ;
        iReturn=hsELMatrixReadTable(ghComm,iNodeID,iTotal,
cReceiveBuff,&iReadLen,&iReturnCode,1000);

        if (iReturn==0)
        {
            strcpy(sDisplay,"Matrix Read Table:");
            for(iLoop=0;iLoop<iReadLen;iLoop++)
            {
                sprintf(sDisplay+strlen(sDisplay),"%02X
",(BYTE)cReceiveBuff[iLoop]);
            }
        }
        else
        {
            strcpy(sDisplay,"Matrix Read Table:Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.1.5 hsELPolling(Read Transaction Data and Event Logs)

Hardware	HDE-100 series	
API	hsELPolling	
Function	int __stdcall hsELPolling(HANDLE hComm, int iELID, int iPrevRecord, stPollList *stRecord, int *iRecord, int * iReturnCode, unsigned int iTimeout);	
Purpose	Retrieves swiped cards and event records from HDE-100 series.	
Arguments	hComm	
	iELID	Device ID
	iPrevRecord	The no. of event records returned by previous hsPollingData (IReturnEvents)
	stRecord	Structure that points to the contents of data to be transmitted. Kindly refer to remark.
	iRecord	The number of data retrieved and transmitted.
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). Recommended time is 10ms and above.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	stRecord structure: char cEventCode[10]; char cDateTime[20]; char cCard[20]; char cDeviceID[10]; char cStatusCode[10];	

Sample	VC Code:
	int wmId, wmEvent;
	switch (message)
	{
	case WM_INITDIALOG:
	hWndListView = GetDlgItem(hDlg, IDC_LIST1);	

```
            SetFocus(hWndListView);
            InitListViewColumns(hWndListView);
            // FALSE since we set focus to hWndTreeView
            return FALSE;
            break;
        case WM_DESTROY:
            break;
        case WM_NOTIFY:
            // Process notification messages.
            switch (((LPNMHDR) lParam)->code)
            {

            // Fill subitems.
            case LVN_GETDISPINFO:
                //OnGetDispInfo((NMLVDISPINFO *) lParam);
                break;

            // Change labels.
            case LVN_ENDLABELEDIT:
                break;
            }
            return 0;
        case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            switch(wmId)
            {
            case IDB_PollingOnce:
                PollingLoop(hDlg);
                break;
            case IDB_ClearList:
                ListView_DeleteAllItems(hWndListView);
                break;
            case IDB_Close:
                EndDialog(hDlg,wmId);
                return TRUE;
                break;
            case 2:
                EndDialog(hDlg,wmId);
                return TRUE;
```

	} break; } return FALSE;

5.2 Read/Write API

5.2.1 hsELReadParameter(Reads System Parameter)

Hardware	HDE-100 series	
API	hsELReadParameter	
Function	int __stdcall hsELReadParamenter(HANDLE hComm, int iELID, unsigned char *cParaData, int *iParaLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Retrieves system parameter of HDE-100 series	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cParaData	Pointer to buffer of received and transmitted data.
	iTableLen	Length of data received
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=4;
	cReceiveBuff[0x0]=0x00;
	cReceiveBuff[0x1]=0x00;	

```
        cReceiveBuff[0x2]=0x13;
        cReceiveBuff[0x3]=0x00;

        iReturn=hsELReadParameter(ghComm,iNodeID,
cReceiveBuff,&iReadLen,&iReturnCode,1000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Read Para:");
            for(iLoop=0;iLoop<iReadLen;iLoop++)
            {
                sprintf(sDisplay+strlen(sDisplay),"%02X
",(BYTE)cReceiveBuff[iLoop]);
            }
        }
        else
        {
            strcpy(sDisplay,"Read Para:Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.2 hsWriteParameter (Set System Parameter)

Hardware	HDE-100 series	
API	hsWriteParameter	
Function	int __stdcall hsELWriteParamenter(HANDLE hComm, int iELID, unsigned char *cParaData, int iParaLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Sets HDE-100 series system parameter	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cParaData	Pointer to data to be transmitted
	iParaLen	Length of data to be transmitted
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). Recommended time is 10ms and above.
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	int iWrittenLen=23;
	cReceiveBuff[0x0]=0x00;
	cReceiveBuff[0x1]=0x00;	

```
        cReceiveBuff[0x2]=0x13;
        cReceiveBuff[0x3]=0x00;

        cReceiveBuff[0x4]=0x00;
        cReceiveBuff[0x5]=0x00;
        cReceiveBuff[0x6]=0xE8;
        cReceiveBuff[0x7]=0x03;

        cReceiveBuff[0x8]=0x00;
        cReceiveBuff[0x9]=0x30;
        cReceiveBuff[0xa]=0x30;
        cReceiveBuff[0xb]=0x30;
        cReceiveBuff[0xc]=0x30;
        cReceiveBuff[0xd]=0x00;
        cReceiveBuff[0xe]=0x00;
        cReceiveBuff[0xf]=0x00;
        cReceiveBuff[0x10]=0x00;
        cReceiveBuff[0x11]=0x00;
        cReceiveBuff[0x12]=0x00;
        cReceiveBuff[0x13]=0x00;
        cReceiveBuff[0x14]=0x00;

        cReceiveBuff[0x15]=0x00;
        cReceiveBuff[0x16]=0x00;
        iReturn=hsELWriteParameter(ghComm,iNodeID,
cReceiveBuff,iWrittenLen,&iReturnCode,1000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Write Para:OK!");
        }
        else
        {
            strcpy(sDisplay,"Write Para:Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.3 hsELInitialize (System initialization)

Hardware	HDE-100 series	
API	hsELInitialize	
Function	int __stdcall hsELInitialize(HANDLE hComm, int iELID, char cInitFlag, int * iReturnCode, unsigned int iTimeout);	
Purpose	HDE-100 series system initialization	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	clnitFlag	Pointer to data to be tranmitted, kindly refer to appendix.
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond).
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	bit0:1 Deletes all card number bit1:1 Clears all tables bit2:1 Deletes all transaction data and event logs bit3:1 Sets parameters in EEPROM to default settings bit4:1 Deletes Blacklist cards only bit5~7 Reserved	

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	int iWrittenLen=12;	

	iReturn=hsELInitialize(ghComm,iNodeID,(char)31,&iReturnCode,10000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Initial RAC2000EL:OK!");
	}
	else
	{
	strcpy(sDisplay,"Initial RAC2000EL:Error!");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

5.2.4 hsELGetInfo (Reads Firmware Version and Other Info.)

Hardware	HDE-100 series	
API	hsELGetInfo	
Function	int __stdcall hsELGetInfo(HANDLE hComm, int iELID, unsigned char *cInfoData, int *iInfoLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Retrieves the firmware version and other information of HDE-100 series.	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cInfoData	Pointer to buffer of retrieved data, kindly refer to appendix.
	iInfoLen	Length of data received
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond).
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	Total of 16 bytes	

	Byte1: Type of version Byte2: Major Version Byte3: Minor Version Byte4: Beta Version Byte5: Year Byte6: Month Byte7: Date Byte8~11:Total number of valid card stored Byte12~15: Total number of transaction data stored

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	iReturn=hsELReadDeviceInfo(ghComm,iNodeID,
	cReceiveBuff,&iReadLen,&iReturnCode,1000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Get Device Info:");
	for(iLoop=0;iLoop<iReadLen;iLoop++)
	{
	sprintf(sDisplay+strlen(sDisplay),"%02X
	",(BYTE)cReceiveBuff[iLoop]);
	}
	}
	else
	{
	strcpy(sDisplay,"Get Device Info:Error:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

5.2.5 hsELAddAuthorization (Add Multiple Cards)

Hardware	HDE-100 series	
API	hsELAddAuthorization	
Function	int __stdcall hsELAddAuthorization(HANDLE hComm, int iELID, int iRecord, struct_CardFormat * stRecord, int * iReturnCode, unsigned int iTimeout);	
Purpose	Adds multiple valid card numbers at a time	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	iRecord	Total number of new cards to be added.
	stRecord	Pointer to the structure with the lists of card numbers to be added. Kindly refer to appendix.
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond).
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	int iType; char cCardNo[12]; int iHoliday; int iTime; unsigned char cActiveFloor[8];	

Sample	VC Code:
	
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;	

```
        int iReadLen=0;
        struct_CardFormat stCard[16];
        memset(stCard,0,sizeof(stCard));

        stCard[0].iType=0x00;
        strcpy(stCard[0].cCardNo,"0000135724");
        stCard[0].iHoliday=0;
        stCard[0].iTime=255;
        memset((char*)stCard[0].cActiveFloor,0xff,8);
        iReturn=hsELAddAuthorization(ghComm,iNodeID,0x01,stCard,
&iReturnCode,1000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Insert Card:OK!");
        }
        else
        {
            strcpy(sDisplay,"Insert Card:Error!");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.6 hsELDeleteAuthorization (Delete Single Authorization)

Hardware	HDE-100 series	
API	hsELDeleteAuthorization	
Function	int __stdcall hsELDeleteAuthorization(HANDLE hComm, int iELID, char *cCardNo, int * iReturnCode, unsigned int iTimeout);	
Purpose	Deletes a single record of authorization	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cCardNo	Pointer to card number to be transmitted
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond).
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	iReturn=hsELDeleteAuthorization(ghComm, iNodeID,"0000135724",&iReturnCode,5000);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Delete Card:OK!");
	}
	else	

	{
	if (iReturnCode==0x06)
	{
	strcpy(sDisplay,"Delete Card:Not Exist!");
	}
	else
	{
	strcpy(sDisplay,"Delete Card:Error!");
	sprintf(sDisplay+strlen(sDisplay),"%04X
	",(WORD)iReturn);
	}
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

5.2.7 hsELQueryAuthorization (Query Single Auhorization)

Hardware	HDE-100 series	
API	hsELQueryAuthorization	
Function	int __stdcall hsELQueryAuthorization(HANDLE hComm, int iELID, char *cCardNo, unsigned char * cCardFormatData,	
Purpose	Queries the authorization of a single card	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cCardNo	Pointer to contents of card number to be transmitted.
	iReturnCode	Error code returned by HDE-100 series Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond).
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];	

```
    memset(sDisplay,0,sizeof(sDisplay));
    unsigned char cReceiveBuff[280];
    memset(cReceiveBuff,0,sizeof(cReceiveBuff));
    int iNodeID=0x01;
    iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
    int iLoop=0;
    int iReadLen=0;
    iReturn=hsELQueryAuthorization(ghComm,
iNodeID,"0000135724",cReceiveBuff,&iReadLen,&iReturnCode,100
0);
    if (iReturn==0)
    {
        if (memcmp(cReceiveBuff+0x01,"0000135724",10)==0)
        {
            strcpy(sDisplay,"Query Card Found:OK!");

        }
        else
        {
            strcpy(sDisplay,"Query Card Not Found:OK!");

        }
    }
    else
    {
        if (iReturnCode==0x06)
        {
            strcpy(sDisplay,"Query Card:Not Exist!");
        }
        else
        {
            strcpy(sDisplay,"Query Card:Error!");
            sprintf(sDisplay+strlen(sDisplay),"%04X
",(WORD)iReturn);
        }
    }

    SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.8 hsELDeleteAllAuthorization (Deletes All Valid Cards)

<table>
<tr><td>Hardware</td><td colspan="2">HDE-100 series</td></tr>
<tr><td>API</td><td colspan="2">hsELDeleteAllAuthorization</td></tr>
<tr><td>Function</td><td colspan="2">int __stdcall hsELDeleteAllAuthorization(
HANDLE hComm,
int iELID,
int * iReturnCode,
unsigned int iTimeout);</td></tr>
<tr><td>Purpose</td><td colspan="2">Deletes all valid card number</td></tr>
<tr><td rowspan="4">Arguments</td><td>hComm</td><td>COM Port or TCP/IP handle value.</td></tr>
<tr><td>iELID</td><td>Device ID</td></tr>
<tr><td>iReturnCode</td><td>Error code returned by HDE-100 series. Kindly refer to appendix 2.1.</td></tr>
<tr><td>iTimeout</td><td>Operation timed out (millisecond). .</td></tr>
<tr><td>Return Value</td><td colspan="2">When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.</td></tr>
<tr><td>Application</td><td colspan="2"></td></tr>
<tr><td>Remark</td><td colspan="2">VC Code:

int iReturn=0;
int iReturnCode=0;
char sDisplay[256];
memset(sDisplay,0,sizeof(sDisplay));
unsigned char cReceiveBuff[280];
memset(cReceiveBuff,0,sizeof(cReceiveBuff));
int iNodeID=0x01;
iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
int iLoop=0;
int iReadLen=0;
iReturn=hsELDeleteAllAuthorization(ghComm,
iNodeID,&iReturnCode,1000);
if (iReturn==0)
{
strcpy(sDisplay,"Delete All Card:OK!");
}
else
{
strcpy(sDisplay,"Delete All Card:Error!");</td></tr>
</table>

	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	
	SetDlgItemText(IDC_EDITReturn,sDisplay);

Sample	

5.2.9 hsELReadDeviceInfo (Retrieve Model Number)

Hardware	HDE-100 series	
API	hsELReadDeviceInfo	
Function	int __stdcall hsELReadDeviceInfo(HANDLE hComm, int iELID, unsigned char *cInfoData, int *iInfoLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Retrieve device's model number	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cInfoData	Pointer to buffer of received data
	iInfoLen	Length of data received
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). .
Return Value	When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;	

```
    int iReadLen=0;
    iReturn=hsELReadDeviceInfo(ghComm,
iNodeID,cReceiveBuff,&iReadLen,&iReturnCode,1000);
    if (iReturn==0)
    {
        strcpy(sDisplay,"Get Device Info:");
        for(iLoop=0;iLoop<iReadLen;iLoop++)
        {
            sprintf(sDisplay+strlen(sDisplay),"%02X
",(BYTE)cReceiveBuff[iLoop]);
        }
    }
    else
    {
        strcpy(sDisplay,"Get Device Info:Error:");
        sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
    }
    SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.10 hsELSetTime (Set Device Time)

Hardware	HDE-100 series	
API	hsELSetTime	
Function	int __stdcall hsELSetTime(HANDLE hComm, int iELID, char *cDate, char *cTime, int * iReturnCode, unsigned int iTimeout);	
Purpose	Set time of HDE-100 series	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cDate	Date Format (YYYYMMDD)+Weekday
	cTime	Time Format (HHMMSS)
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	cDate includes weekdays Ex: 2007/02/10 Monday cDate = 200702101-the last digit 1 represents Monday and so on.. cTime = 120000	

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	char cDate[20];
	char cTime[20];
	memset(cDate,0,sizeof(cDate));
	memset(cTime,0,sizeof(cTime));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iPos=0;
	SYSTEMTIME st;	

```
        GetLocalTime(&st);
        iPos=0;
        sprintf(cDate+iPos,"%04d",st.wYear);
        iPos=iPos+4;
        sprintf(cDate+iPos,"%02d",st.wMonth);
        iPos=iPos+2;
        sprintf(cDate+iPos,"%02d",st.wDay);
        iPos=iPos+2;
        sprintf(cDate+iPos,"%01d",(st.wDayOfWeek));

        iPos=0;
        sprintf(cTime+iPos,"%02d",st.wHour );
        iPos=iPos+2;
        sprintf(cTime+iPos,"%02d",st.wMinute );
        iPos=iPos+2;
        sprintf(cTime+iPos,"%02d",st.wSecond );

        iReturn
=hsELSetTime(ghComm,iNodeID,cDate,cTime,&iReturnCode,1000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Set DateTime:OK.");
        }
        else
        {
            strcpy(sDisplay,"Set DateTime: Error Number:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.11 hsELGetTime (Reads Device Time)

Hardware	HDE-100 series	
API	hsELGetTime	
Function	int __stdcall hsELGetTime(HANDLE hComm, int iELID, char *cDate, char *cTime, int * iReturnCode, unsigned int iTimeout);	
Purpose	Retrieves time from HDE-100 series.	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cDate	Date Format (YYYYMMDD)+Weekday
	cTime	Time Format (HHMMSS)
	iReturnCode	Error code returned by HDE-100 series.. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	cDate includes weekdays Ex: 2007/02/10 Monday cDate = 200702101-the last digit 1 represents Monday and so on.. cTime = 120000	

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	char cDate[20];
	char cTime[20];
	memset(cDate,0,sizeof(cDate));
	memset(cTime,0,sizeof(cTime));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	
	iReturn	

	=hsELGetTime(ghComm,iNodeID,cDate,cTime,&iReturnCode,500);
	if (iReturn==0)
	{
	strcpy(sDisplay,"Device Time:");
	strcat(sDisplay,cDate);
	strcat(sDisplay," ");
	strcat(sDisplay,cTime);
	}
	else
	{
	strcpy(sDisplay,"Get DateTime: Error Number:");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);
	}

5.2.12 hsELReleaseAlarm (Deactivate Alarm)

Hardware	HDE-100 series.	
API	hsELReleaseAlarm	
Function	int __stdcall hsELReleaseAlarm(HANDLE hComm, int iELID, char cAlarmFlag, int * iReturnCode, unsigned int iTimeout);	
Purpose	Sends a command to deactivate alarm	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	cAlarmFlag	Pointer to data to be transmitted, kindly refer to appendix.
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	cAlarmFlag bit0:1 Deactivates alarm for blacklist card bit1:1 Deactivates reswiped card alarm bit2-7 Reserved	

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	int iWrittenLen=12;
	iReturn=hsELReleaseAlarm(ghComm,iNodeID,(char)31,
	&iReturnCode,10000);	

	if (iReturn==0)
	{
	strcpy(sDisplay,"Release Alarm RAC2000EL:OK!");
	}
	else
	{
	strcpy(sDisplay,"Release Alarm RAC2000EL:Error!");
	sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
	}
	SetDlgItemText(IDC_EDITReturn,sDisplay);

5.2.13 hsELPublicFloor (Set Public Access Floor)

Hardware	HDE-100 series.	
API	hsELPublicFloor	
Function	int __stdcall hsELPublicFloor(HANDLE hComm, int iELID, int iTotal, unsigned char *cFloorData, int *iReadLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Defines the public access floor	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	iTotal	Total number of public access floors
	cFloorData	Pointer to data to be transmitted, kindly refer to appendix.
	iReadLen	Length of data received
	iReturnCode	Error code returned by HDE-100 series. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	cFloorData：8 Bytes Byte 1 bit0 First Floor Byte 1 bit1 Second Floor Byte 1 bit2 Third Floor . . . . . . Byte 8 bit6 63rd Floor Byte 8 bit7 64th Floor 0-> Disable Public Access Floor 1-> Enable Public Access Floor	

Sample	VC Code:
	int iTotal=0;
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];	

```
        memset(sDisplay,0,sizeof(sDisplay));
        unsigned char cReceiveBuff[280];
        memset(cReceiveBuff,0,sizeof(cReceiveBuff));
        int iNodeID=0x01;
        iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
        int iLoop=0;
        int iReadLen=0;
        memset(cReceiveBuff,0x00,sizeof(cReceiveBuff));
        iTotal=8;
        cReceiveBuff[0]=0xff ;
        cReceiveBuff[1]=0xfe ;
        cReceiveBuff[2]=0xff ;
        cReceiveBuff[3]=0xff ;
        cReceiveBuff[4]=0xff ;
        cReceiveBuff[5]=0xff ;
        cReceiveBuff[6]=0xff ;
        cReceiveBuff[7]=0xff ;

        iReturn=hsELPublicFloor(ghComm,iNodeID,iTotal,
cReceiveBuff,&iReadLen,&iReturnCode,1000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Set Public Floor : OK!");
        }
        else
        {
            strcpy(sDisplay,"Set Public Floor : Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

5.2.14 hsELSetReader (Set Slave Reader)

Hardware	HDE-100 series	
API	hsELSetReader	
Function	int __stdcall hsELSetReader(HANDLE hComm, int iELID, int iReaderID, int iIndex, unsigned char * cSendData, int iSendDataLen, int * iReturnCode, unsigned int iTimeout);	
Purpose	Set slave device of HDE-100.	
Arguments	hComm	COM Port or TCP/IP handle value.
	iELID	Device ID
	iReaderID	Slave reader's ID
	iIndex	Reader's Read/Write index
	cSendData	Parameter to set to reader
	iSendDataLen	Length of send data
	iReturnCode	Error code returned by HDE-100 series.. Kindly refer to appendix 2.1.
	iTimeout	Operation timed out (millisecond).
Return Value	When return "HF_RET_SUCESS" is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	VC Code:
	int iReturn=0;
	int iReturnCode=0;
	char sDisplay[256];
	memset(sDisplay,0,sizeof(sDisplay));
	unsigned char cReceiveBuff[280];
	memset(cReceiveBuff,0,sizeof(cReceiveBuff));
	int iNodeID=0x01;
	iNodeID=GetDlgItemInt(IDC_EDITID,NULL);
	int iLoop=0;
	int iReadLen=0;
	memset(cReceiveBuff,0x00,sizeof(cReceiveBuff));	

```
        cReceiveBuff[0x0]=0x00;
        cReceiveBuff[0x1]=0x00;
        cReceiveBuff[0x2]=0x0D;
        cReceiveBuff[0x3]=0x0F;
        cReceiveBuff[0x4]=0x01;
        cReceiveBuff[0x5]=0x02;
        cReceiveBuff[0x6]=0x00;

        iReturn=hsELSetReader(ghComm,iNodeID,1,23,
cReceiveBuff,7,&iReturnCode,2000);
        if (iReturn==0)
        {
            strcpy(sDisplay,"Set Reader : OK!");
        }
        else
        {
            strcpy(sDisplay,"Set Reader : Error:");
            sprintf(sDisplay+strlen(sDisplay),"%04X ",(WORD)iReturn);
        }
        SetDlgItemText(IDC_EDITReturn,sDisplay);
```

Chapter 6 ECU-680 Series

6.1 Read/Write API

6.1.1 hsECUReadIO(Read I/O Status)

Hardware	ECU-680 series.	
API	hsECUReadIO	
Function	int __stdcall hsECUReadIO(HANDLE hComm,int iECUID,int *cSensor,int *cRelay,int *iReturnCode,unsigned int iTimeout)	
Purpose	Read I/O status	
Arguments	hComm	COM Port or TCP/IP handle value.
	iECUID	Device ID
	cSensor	Sensor status retrieved.
	cRelay	Relay value retrieved.
	iReturnCode	Reserved
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	Sensor Status: Bit 0 : Sensor 1 status, 1→ Close 0→ Open Bit 1 : Sneosr 2 status, 1→ Close 0→ Open Bit 2~7 : Reserved Relay Status: Bit 0 : Relay 1, 1→Activate, 0→Inactivate Bit 1 : Relay 2, 1→Activate, 0→Inactivate Bit 2~7 : Reserved	
Sample		

6.1.2 hsECUReadParameter(Reads All Parameter Values)

Hardware	ECU-680 series.	
API	hsECUReadParamenter	
Function	int __stdcall hsECUReadParamenter(HANDLE hComm,int iECUID,unsigned char *cParaData,int *iParaLen,int *iReturnCode,unsigned int iTimeout)	
Purpose	Reads all parameter values	
Arguments	hComm	COM Port or TCP/IP handle value.
	iECUID	Device ID
	cParaData	Parameter contents retrieved
	iParaLen	Parameter length retrieved
	iReturnCode	Reserved
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		
Sample		

6.1.3 hsECUReadPower(Reads Power Status (with Card No))

Hardware	ECU-680 series.	
API	hsECUReadPower	
Function	int __stdcall hsECUReadPower(HANDLE hComm,int iECUID,char *cPower,unsigned char *cCardNo,int * iReturnCode,unsigned int iTimeout)	
Purpose	Reads power status with card number.	
Arguments	hComm	COM Port or TCP/IP handle value.
	iECUID	Device ID
	cPower	Power status retrieved
	cCardNo	Retrieve current using card number
	iReturnCode	Reserved
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	When power on and cCardNO is empty, it means the card has been drawn out. System will process into Extend Power Off Time Function.	
Sample		

6.1.4 hsECUAddCard (Adds a Card)

Hardware	ECU-680 series.	
API	hsECUAddCard	
Function	int __stdcall hsECUAddCard(HANDLE hComm,int iECUID,char *cCardNo,int iCardLen,int iTimeZone,int Validity,int ValiditYear,int ValidityMouth,int ValidityyDay,int ValidityHour,int ValidityMinute , unsigned int iTimeout)	
Purpose	Add a valid card.	
Arguments	hComm	COM Port or TCP/IP handle value.
	iECUID	Device ID
	cPower	Power status retrieved
	cCardNo	Retrieve current using card number
	iReturnCode	Reserved
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark	When power on and cCardNO is empty, it means the card has been drawn out. System will process into Extend Power Off Time Function.	
Sample		

6.1.5 hsECUPolling (Polling Device)

Hardware	ECU-680 series.	
API	hsECUPolling	
Function	int __stdcall hsECUPolling(HANDLE hComm,int iECUID,int iPrevRecord,stPollList *stRecord,int *iRecord,int * iReturnCode,unsigned int iTimeout)	
Purpose	Sends polling command to device and wait for the returned data.	
Arguments	hComm	COM Port or TCP/IP handle value.
	iECUID	Device ID
	iPrevRecord	No. of records previously retrieved
	stRecord	Structure that points to the contents of data retrieved.
	iRecord	No. of records retrieved
	iReturnCode	Reserved
	iTimeout	Operation timed out (millisecond). .
Return Value	When return “HF_RET_SUCESS” is true. Kindly refer to Chapter 3 for other return value.	
Application		
Remark		

Sample	typedef struct stPollList
	{
	char cEventCode[5]; //Event Code
	char cDateTime[20]; //Date Time
	char cCard[20]; //Card Number
	char cDeviceID[10]; //Device ID
	char cReaderID[10]; //Reader ID
	} stPollList;	

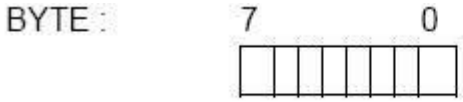
Chapter 7 Appendix

Appendix 1: RAC-2000 series Description

1.1 Basic Format Description

RAC-2000 ID range : 1 – 255

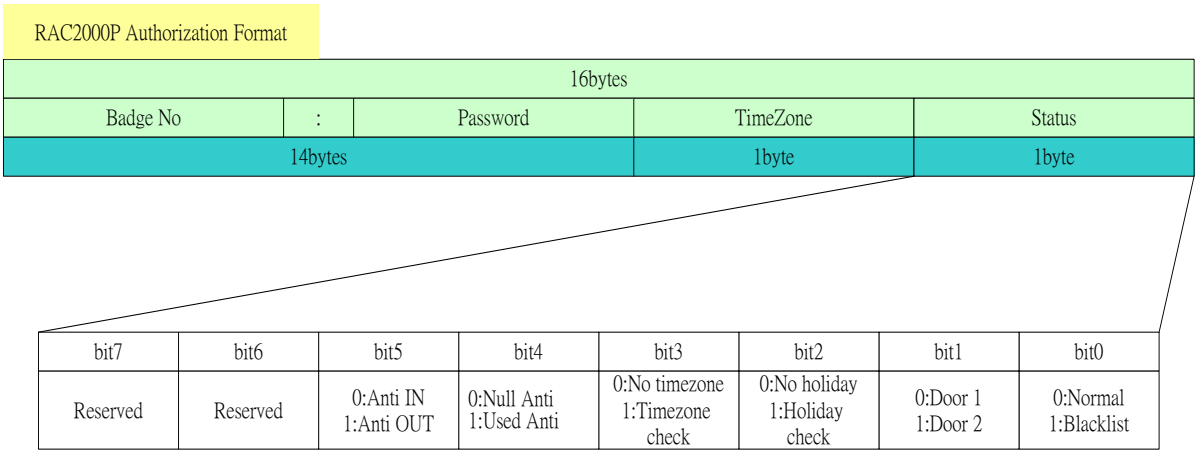
System byte indicates:



DOOR1:Reader1 ID : 0001
Reader2 ID : 0002

DOOR2: Reader1 ID : 0003
Reader2 ID : 0004

1.2 RAC-2000 series Valid Card Format



bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Reserved	Reserved	0:Anti IN 1:Anti OUT	0:Null Anti 1:Used Anti	0:No timezone 1:Timezone check	0:No holiday 1:Holiday check	0:Door 1 1:Door 2	0:Normal 1:Blacklist

Note:

1. Suit for RAC-2000P/PV/PS/PN/PSN/WS/WSN and HDP-100/100S.
2. Password: Appends FF if insufficient or no password is provided.
3. The password is compressed by combining 2 digits into 1 byte.
4. ":", 【Colon】is used to separate the card number from the password.

1.3 RAC-2000G Valid Card Format

RAC2000GAuthorization Format

8bytes		
Badge No	Password	Status
5bytes	2bytes	1byte

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Reserved	Reserved	0:Anti IN 1:Anti OUT	0:Null Anti 1:Used Anti	Reserved	Reserved	0:Door 1 1:Door 2	Reserved

Note:

1. Suit for RAC-2000G only.
2. Password: Appends FF if insufficient or no password is provided.
3. The card number and password are compressed by combining 2 digits into 1 byte.

1.4 RAC-2000 series Swiped Card Records

History Log Format

19bytes				
Len	Status1	Status2	Datetime	Badge No
1byte		1byte	4bytes	1-13 byte

Length: 4bit indicates the length of the card number

Status 1: 4bit

bit0 = 0: requires input of card number

=1 swipe card

Bit1,2 =00 reader 1

=01 reader 2

=10 reader 3

=11 reader 4

Bit3 = 0 Event

=1 Event with event code but without card number. Event code is located at the 7th and 8th byte.

Card number saves data according to length

Status 2:

0 = Swipe of valid card
1 = Master card
2 = Release code
3 = Duress card
4 = Duress code
5 = Temporary card
6 = Black list
10 = Guest card
11 = Guest card(unlimited usage)
20 = Card number missing
61 = Swipe of valid card + correct input of password
62 = Swipe of valid card + incorrect input of password
63 = Input of card number + password
67 = Anti-passback error

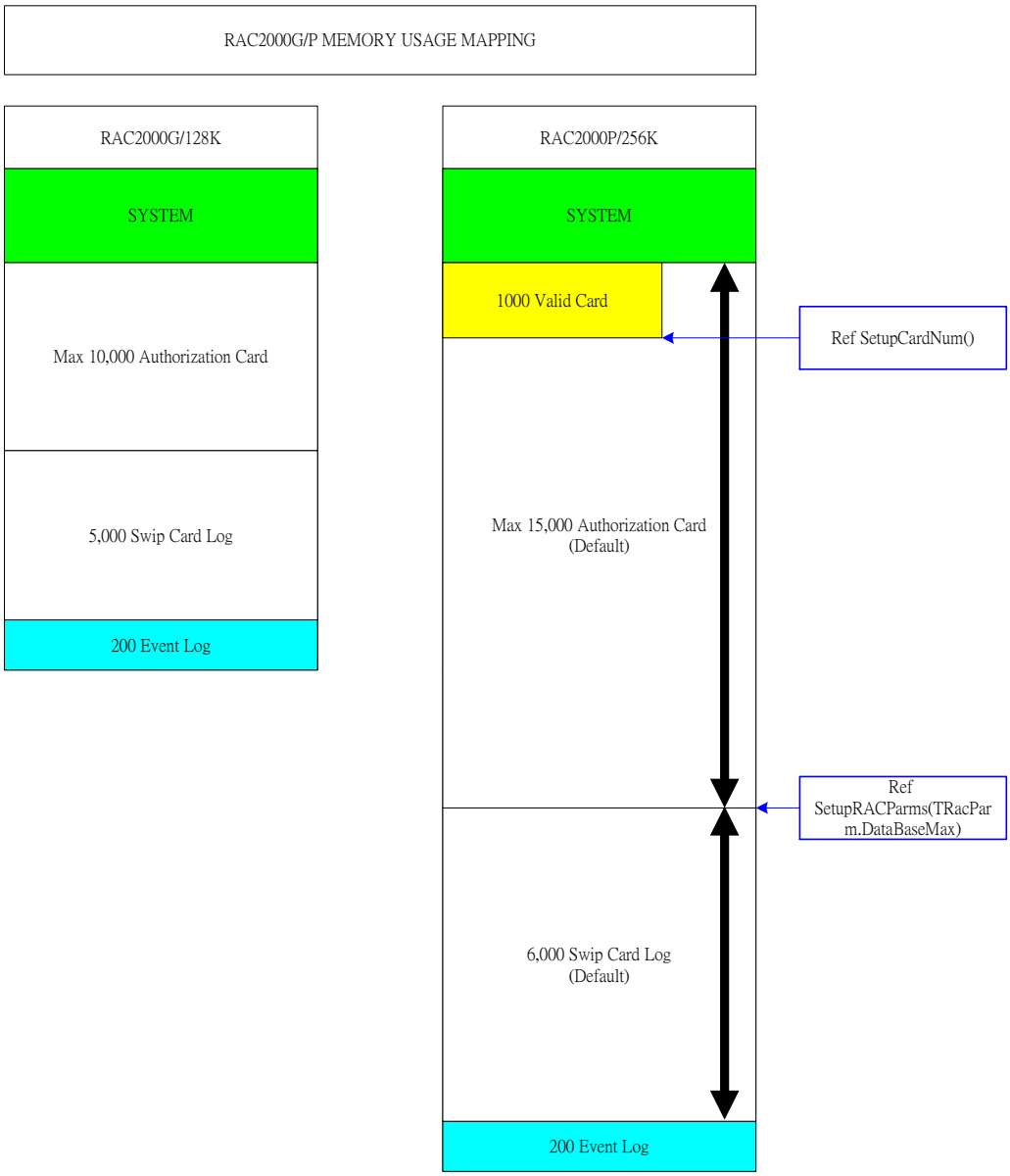
Date and time: Total of 4 byte(transmission starts with Lo byte)

0-5 bit contains seconds

6-11 bit contains minute

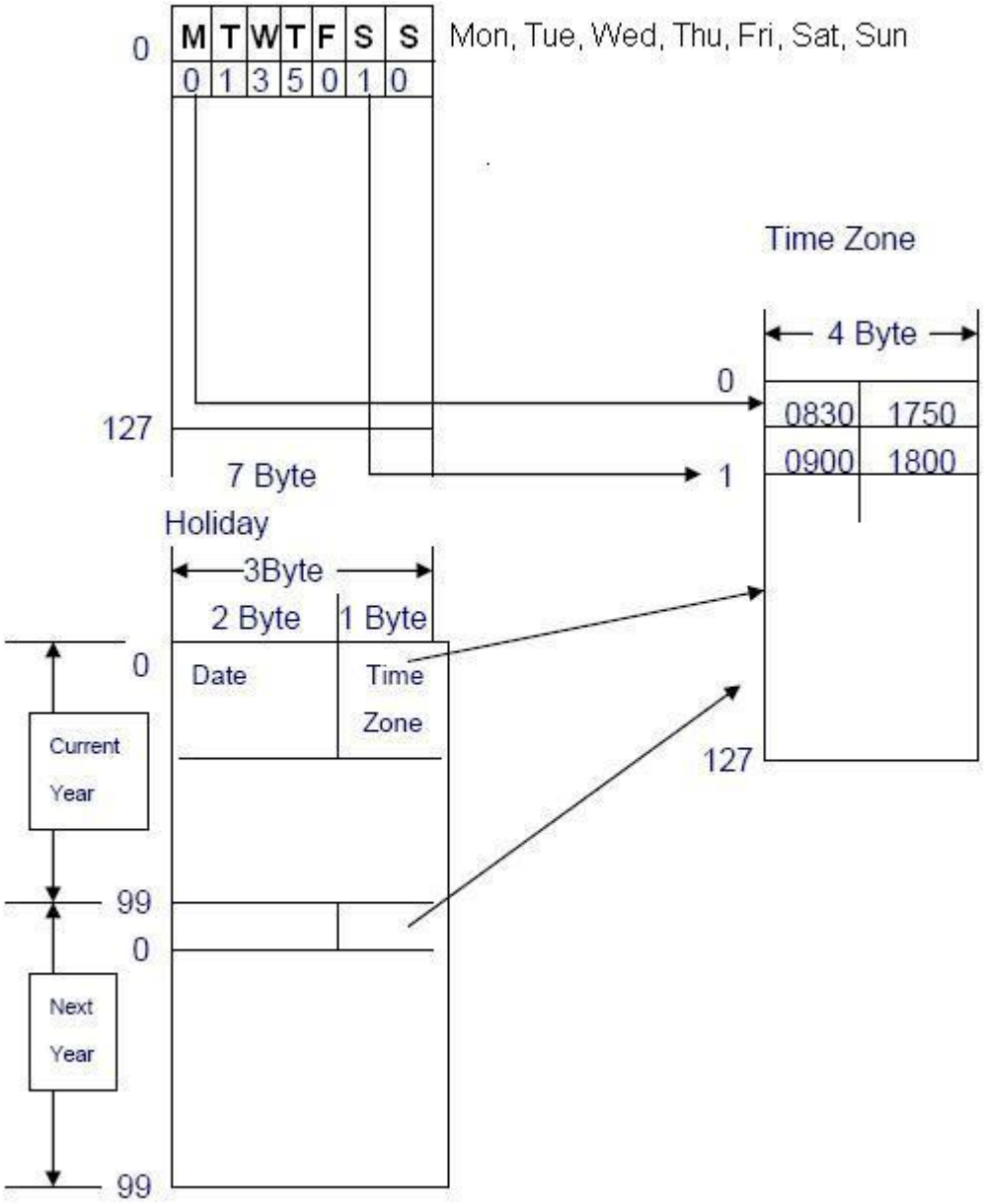
12-16 contains hours
17-21 contains date,
22-25 contains month
26-31 contains year (2000+N)

1.5 Memory Allocation



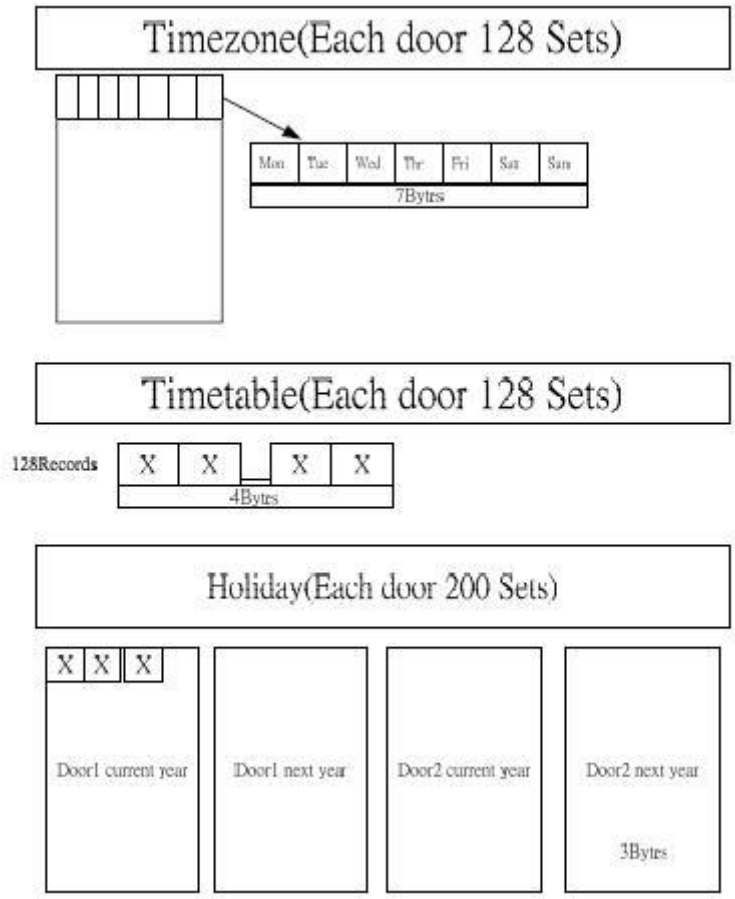
1.6 Time Schedule

Time Schedule, Time Zone, Holiday



※ NOTE: FF is appended if time zone, time schedule and holiday schedule is not provided.

1.7 Holiday Timetable



※ NOTE: FF is appended if time zone, time schedule and holiday schedule is not provided.

1.8 Vistor Card Format

RAC-2000 RMA Usaged

Guest Card (50 records)										
		Date Range			Time Range		Week			
Card No	Compare	Y	M	D	Y	M	D			M ~ S	Time	Anti	Reserved
14 bytes	2 bytes	3 bytes			3 bytes			2 bytes	2 bytes	1 byte	2 bytes	1 byte	2 bytes

1.9 Event Code's Return Value (Sensor/Relay)

0120 H	Duress code recognized
0122 H	Disarm code deactivates alarm
0123 H	Master card(code) modified
0301 H	Tamper → Case Sensor
0306 H	Cold startup
0307 H	Reset
0311 H	Door forced open → Door Sensor (Door has been breakthrough)
0312 H	Door prop alarm
0313 H	Deactivate door prop alarm
0318 H	Door opened → Valid access
0319 H	Door opened → Door relay closed
031A H	Exit button pressed → Push button
0401 H	Memory full
041B H	System initialization

※ NOTE: The RAC-2000 series memory allocation above is set for one door only except for the valid card authorization where it is shared to both doors.

1.10 EEPROM Memroy Allocation

Length	ADD	Description	Default	Remarks
2	0000H	EEPROM test	5AH,A5H	Startup Test
2	0002H	Device ID	‘R’, ‘2’	G=R1,P=R2
2	0004H	Retrieve valid vode	01H, 00H	GetCard_CODE
2	0006H	Compare valid vode	00H, 00H	CompCard_CODE
14	0008H	Master card (FFH is appended to unused space)	‘30191000’	MasterCard_CODE
14	0016H	Duress card (FFH is appended to unused space)	1190	ForceCard_CODE
1	0024H	Duress code is required after swiping duress card bit0,4, 0=NO, 1=YES Swipe card after inputting duress code bit1, 5, 0=NO(1,2,4,8 Relay 1,2,3,4)	00H	ForcePin_CODE
1	0025H	Relay will activate afterswiping duress cord. 0=NO(1,2,4,8 Relay1,2,3,4)	82H	ForceRelay_CODE
14	0026H	Duress Code	1190	ForceCode_CODE
1	0034H	Door1 action time	04H	Door1Time_CODE
1	0035H	Door2 action time	04H	Door2Time_CODE
14	0036H	Disarm code (FFH is appended to unused space)	‘0000’	ReleaseCode_CODE
16	0044H	8 sets conditional unlock door schedule x 2 door	FFH	SustainedOpenBlock_CODE
2	0054H	Year of the holiday schedule	0	Holiday year, each door occupies one byte. For year 2011, 0x11,0x11 should be written.
2	0056H	The number of valid cards in SRAM	0	DataBasePoint_CODE(Hi,Lo)
2	0058H	Max. valid card number	10000	DataBaseMax_CODE(Hi,Lo)
1	005AH	Default ID(If DIP SW = 0)	1	NodeID_CODE
1	005BH	Baudrate 1-6=1200,2400,4800,9600,19200,38400	5	BaudRateSet_CODE
1	005CH	Reserved	0	RAC2000P=0,RAC2000G=1
1	005DH	Activate relay for black list. bit0, 4:0 NO,1 YES	11H	BlacklistRelay_CODE
1	005EH	Disarm code with card number 0=NO, 1=YES	0	ReleaseWithCard_CODE
1	005FH	Door Sensor will activate alarm	0	DoorSenserMode_CODE

		bit0,4 = 1 Reader Buzzer buzzes bit1,5 = 1 Alarm Relayactivates		
1	0060H	Door Sensor1 detection time (sec)	0	DoorSenser1Sec_CODE
1	0061H	Door Sensor2 detection time (sec)	0	DoorSenser2Sec_CODE
1	0062H	bit0=1 Memory 90% full alarm bit1=1 Don't overwrite saved records bit2=1 activate alarm relay when memory is full	0	HistoryMemorySet_CODE
1	0063H	Max. failure of swipe card	0	ErrorTime_CODE
1	0064H	The time for the system to cease operation upon reaching maximum number of failed log in attempts.	0	ErrorHaltTime_CODE
1	0065H	Reverse sensor Input 0 = NO, 1=YES bit0,1,2,3=Case1,2,3,4 bit4,6=PushButton1,2 bit5,7=DoorSensor1,2 bit0,1=Case1,2 bit4,6=PushButton1,2 bit5,7=DoorSensor1,2	0	SensorAction_CODE
1	0066H	Case Sensor selection of whether to activate alarm or not 1 Byte bit0,4 = 1 Reader Buzzer buzzers, bit1,5 = 1 Alarm Relay activates bit0,4 = 1 Reader Buzzer buzzers bit1,5 = 1 Alarm Reader beeps	0	CaseSensorMode_CODE
1	0067H	Reserved	0	
1	0068H	0= Anti deactivated 1=Anti By Door Activates Door1 Relay IN/OUT Reader1-2 IN,Reader3-4 OUT 2=Anti By Reader (2000P only) Door1 Relay IN/OUT Reader1 IN,Reader2 OUT Door2 Relay IN/OUT Reader3 IN,Reader4 OUT 3=Anti By Door(2000P only) Door1 Relay IN, Door2 Relay OUT Reader1-2 IN,Reader3-4 OUT	0	AntiMode_CODE*

1	0069H	The time (in Sec.) to check reswipe 0=Disable; 1~255=1~255 seconds	0	
1	006AH	Reserved	10	
1	006BH	Reserved	10	
1	006CH	bit0=Two door interlocking 0=Disable,1=Enable bit1 = One door control, one user at a time 0=Disable,1=Enable	0	
1	006DH	Reserved	0	
1	006EH	Reserved	0	
1	006FH	Reader Status bit0=Reserved bit1=Reserved bit2=0→Disable Number keys 1→Enable Number keys bit3=0→Disable Function Keys 1→Enable Function Keys bit4=Reserved bit5=Reserved bit6=Reserved bit7=0→Disable Case Sensor 1→Enable Case Sensor	7FH	
1	0070H	Door Relay Restoration time setting 0=Door Relay duration over to close Door Relay 1=DOOR SENSOR restoration to close Door Relay	0	
1	0071H	Not check time schedules on slave reader 0=Check time schedules 1=Not check time schedules	0	
1	0072H	Reserved	60	
1	0073H	Reserved	0	
1	0074H	Pad '00" in front of 8 digits of Wiegand 26 Bits card number to become 10 digits 1=pad "00"	0	
1	0075H	If no valid cards, 0=Swipe any card to unlock, 1=Not to unlock	0	
5	0076H	76H-7AH Reserved	FFH	
1	007BH	Whether to activate parity check on	01H	

		Wiegand reader (00H means Not to check; other values mean to check)		
1	007CH	Reserved	60	
3	007DH	7DH-7FH	FFH	
8	0080H	Reserved	FFH	
8	0088H	Codes for password control time schedules		
16	0090H	Each door has 8 time-schedule codes for password control, corresponding to 0090H~0097H: Door 1 0098H~009FH: Door 2		Not support to cross over midnight (Cross-over-midnight has to be divided into two time schedules, e.g. 2100~2359, 0000~0700) The first Byte each door is 0XFF for regular password control; Or the password only required within the time schedules
1	00A0H	Setting (Adjustment) of daylight saving time Bit6~Bit0 =0 Disable =1 Before Adjustment =2 After Adjustment (all other values regarded as Disable) Bit7: Status of adjustment (Please do not change this Bit status) =0 Not yet to adjust =1 Adjusted	00H	
4	00A1H	Start Month/Day/Hour/Minute for daylight saving time (MMDDhhmm in BCD format)	All 00H	
4	00A5H	End Month/Day/Hour/Minute for daylight saving time (MMDDhhmm in BCD Format)	All 00H	
2	00A9H	Adjustment of daylight saving time hhmm (in BCD format)	All 00H	
	00ABH	ABH-F9H Reserved	FFH	
6	00FAH	FAH-FFH For system use only, please do not change it		

128*7	0100H	Time schedule 128sets*7byte (Door1)		TimeBlock1
128*7	0480H	Time schedule 128sets*7byte (Door2)		TimeBlock2
128*4	0800H	Time schedule 128sets*4byte (Door1)		TimeTable1
128*4	0A00H	Time schedule 128sets*4byte (Door2)		TimeTable2
200*3	0C00H	Holiday schedule 200sets*3byte (Door1)		Holyday1
200*3	0E58H	Holiday schedule 200sets*3byte (Door2)		Holyday2
	10B0H	10B0H-10FFH for system use only. Please don't modify it		Blank2
	1100H	1100H-19FFH Reserved		
480	1A00H	Wiegand decode setting		(RAC-2000WS/RAC-2000WSN Only)
	1BE0H	1BE0H-1FFFH Reserved		

Wieganddecode setting (Address 1A00H~1BDFH:Total 480Bytes)

Decode settings for all kinds of number of bits; Each kind includes 10 Bytes, Status ,1 Byte, each bit Enable / Disable status, 9 Bytes

Status 1 Byte :

bit 0 whether to decode 1=Enable / 0=Disable

bit 1 : rules for card number to decode

1=Special rules to decode, increase according to requirements

0=Gerneral rules to decode, get max. 32 bits and convert to 10 digits card number (decimal)

bit 2-7 = 0 (Reserved)

bit Enable/Disable Status 9Byte : all initial values are all FFh , (1=Enable / 0=Disable)

11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111

bit1 ……bit72

The bellow table shows initial values of decoding for all kinds of number of bits

	Status	Enable/Disable (bit1 – bit72)
26 Bit	03h	7Fh,FFh,FFh,80h,00h,00h,00h,00h,00h
27 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
28 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
29 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
30 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
31 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
32 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh

33 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
34 Bit	01h	7Fh,FFh,FFh,FFh,80h,00h,00h,00h,00h
35 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
36 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
……	…….	……..
72 Bit	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh
Reseerved	00h	FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh,FFh

1.11 Anti-passback Settings

To activate anti-passback, use hacSetEEData's function to write the parameter to address 0068H.

ANTI(4Bit)

AntiMode = 1 By Door Activates Door1 Relay IN/OUT Reader1-2 IN,Reader3-4 OUT AntiMode = 3 By Door Door1 Relay IN, Door2 Relay OUT Reader1-2 IN,Reader3-4 OUT	AntiMode = 2 By Reader Door1 Relay IN/OUT Reader1 IN Reader2 OUT	AntiMode = 2 By Reader Door2 Relay IN/OUT Reader3 IN Reader4 OUT
bit4: 0:Anti default value 1: Card no. is unavailable	bit4: 0:Anti default value 1: Card no. is unavailable	bit4: 0:Anti default value 1: Card no. is unavailable
bit5: 0: Previous swipe: IN 1: Previous swipe: OUT	bit5: 0: Previous swipe: IN 1: Previous swipe: OUT	bit5: 0: Previous swipe: IN 1: Previous swipe: OUT

1.12 RAC-2000G Memory Allocation

Address	Length	Purpose
000000H-00001FH	32	Cache index
000020H-00065FH	1600	Guest card (50 records*32 byte)
000660H-000C9FH	1600	Saves event (200 records*8 byte)
000CA0H-01451FH	10000	Valid cards (10000 records*8 byte)
014520H-01FFFFH		Swipe card records

Note:

1. Suit for RAC-2000G only.

1.13 RAC-2000 series Memory Allocation

Address	Length	Purpose
000000H-00001FH	32	Cache index
000020H-00065FH	1600	Guest card (50 records*32 byte)
000660H-000C9FH	1600	Saves event (200 records*8 byte)
000CA0H-027D9FH	10000	Valid cards (10000 records*16 byte)
027DA0H-03FFFFH		Swipe card records

Note:

1. Suit for RAC-2000P/PV/PS/PN/PSN/WS/WSN and HDP-100/100S.

Appendix 2: HDE-100 series Description

2.1 Error Code List

Hexadecial Code	Descrition
0001	Allocated memory space for card nmber exceeded. Length of valid card number mismatched.
0002	Invalid length of card number
0003	Exceeded the max number of valid cards
0006	Card number does not exist
0007	Data length transmitted is too long
0008	Data length retrieved is too long
0009	Invalid data length
000F	Error in the data transmitted to HDE-100 series

2.2 EEPEOM Memory Allocation

Length	ADD	Description	Remarks
1	0	Overwrite swiped card records	0 — No alert (default) 1 — When memory reaches 90% full, reader beeps.
1	1	Open door on absence of valid card?	0 — Lock door 1 — Open door (default)
2	2	Elevator keypad activation time (Lo, Hi)	byte(Lo,Hi), Default is 10 ms.
1	4	Unrestricted time zone index	Reserved
12	5	Disarm code	When disarm code did not reach 10 byte(default:0000). Disarm code deactivates: 1. Blacklist alarm 2. Reswiped card alarm
1	17	Reswipe card alarm	Actuvate/Deactivate alarm 0 -- Disable alarm (default) 1 – Enable alarm, alarm output time duration.
1	18	Reswipe card delay time	0 – System ignores this setting(default) 1 ~ 255 – time duration in which swiping card twice is considered a reswipe of card.

8	19	Public accessible floor	64 bit indicating the 64 floor levels controlled by HDE-100 series.
1	27	24 hours open/close	Reserved
12	28	Master Card	a. 24 hours – all floors are accessible. b. Deactivate alarm c. Deactivate temporary restriction of card swipe Master card can only perform the settings above one at a time.
1	40	Set password Mode (Reserved)	
1	41	Workday password time schedule index (reserved)	
1	42	Holiday password time schedule index (reserved)	
		Reserved	

2.3 Data Format

2.3.1 Card Format

Card Type(1)	Card No.(12)	Holiday (1)	Time Schedule (1)	Floor Level(8)

● Card Tyep : 1 byte

Description:

0x00: Valid Card
Checks card validity and record the result
0x01: Activate alarm for blacklist card
Enable alarm and record the event as trigger alarm for blacklist card

● Card No. : 12 byte. Umcompressed, appends 0 for card number less than 12 byte.

● Holiday:

bit0: 0 → Unrestricted Holiday
1 → Restricted Holiday
Bit 1~7 Reserved

● Time Schedule: Time schedule reference, 0 signifies access denied
255 signifies access granted.

● Floor Level: Accessible floor level upon valid card swipe. Max. of 64 floor levels can be

controlled as 8*8=64 bit

If bit = 0 → Restricted floor level

1 → Accessible floor level

Byte1, Byte2, Byte3, Byte4, Byte5, Byte6, Byte7, Byte8,

Byte 1 bit 0 1st floor

Byte 1 bit 1 2nd floor

Byte 1 bit 2 3rd floor

Byte 1 bit 3 4th floor

……

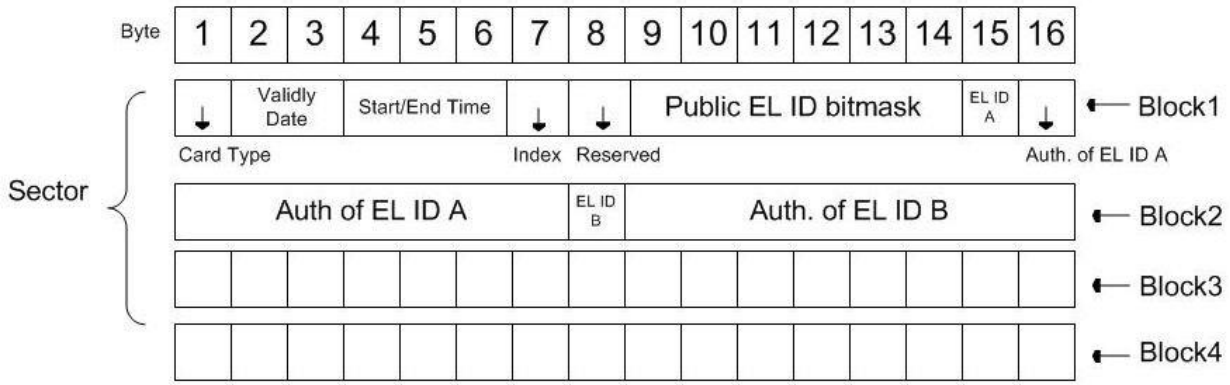
Byte 8 bit 0 57th floor

……

Byte 8 bit 7 64th floor

2.3.2 Valid Card Format

Standard Mifare Card Plan, Version 1.0



Block 1:

- Card Tyep (byte 1) : 03 → Valid card

  04 → Guest card

- Validly Date and time start/end:

Validly Date: byte 2, byte 3, Time starat and time end : byt 4, byt 5, byt 6

Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
bit 7…bit0	bit7….bit0	bit 7….bit0	bit7….bit0	bit7….bit0
0000000 0	000 00000	00000 000	000 00000	000000 00
Year	Month Date	Hour	Minute	Hour Minute
Validy Date		Time Start		Time End

※ When byte2 and byte3 for date start is 0xff, 0xff, then the card has unlimited validity.

• Index: Byte 7

The index for next sector, 0xff means do not have next sector. If has next sector, the new idex is in 1st byte of frone three blcoks' authorization. The 2nd bytes format is 1bytes EL ID + 8 bytes Floors.

• Public EL ID bitmask: byte 9-14

When bit=0 → The elevator can not open public floors

bit=1 → The elevator can open public floors

Byte 9, bit0 Elevator ID 0

Byte 9, bit1 Elevator ID 1

Byte 9, bit2 Elevator ID 2

Byte 9, bit3 Elevator ID 3

…….

Byte 14, bit0 Elevator ID 46

Byte 14, bit7 Elevator ID 47

• EL ID A: byte 15

When HDE-100 series's ID is same with elevator A 's ID, user will have authorization of Elevator A.

• Auth. EL ID A: byte 16, block 2 byte 1-7

Enable open floors. Byte1 bit0 is lower floor.Byte7 bit7 is highest floor

• EL ID B: byte 15

When HDE-100 series's ID is same with elevator B's ID, user will have authorization of Elevator B.

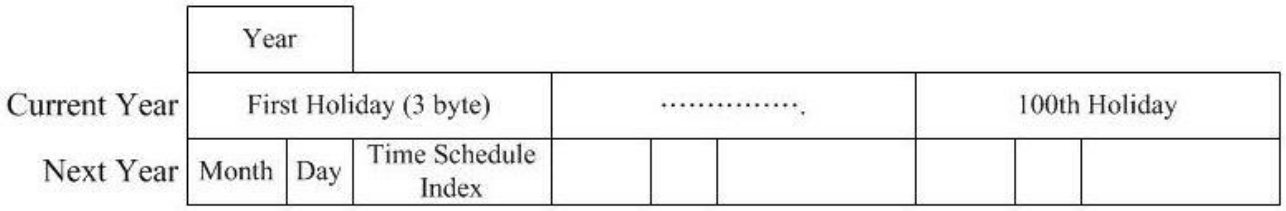
• Auth. EL ID B

Enable open floors. Byte1 bit0 is lower floor.Byte7 bit7 is highest floor

• Block 3-4: Reserved

2.3.3 Holiday Index

(Address 0: Total:601 bytes)



.

● Year (1 byte):
Indicates the current year.
Example : Year 2010=>0x10 (BCD format)

● Time Schedule index: 0 → signifies no time shchedule set
1~128 → references to the set time schedule
255 → signifies all day access

※ Note:
1. Month and date are in BCD format, example: 12/1 ➔ 0x12, 0x01
2. Each index can have 100 holidays. When you place 0xFF,0xFF in month and date, it signifies no comparison of authorization needed.

2.3.4 Time Schedule Format

(Address 601 : Total 1778 bytes)

Index	Weekday Mon	Tue	Wed	Thu	Fri	Sat
1	Time Schedule Index					
2						
..						
..						
254						

● Time Schedule index: 0 → signifies no time shchedule set
1~128 → references to the set time schedule
255 → signifies all day access

2.3.5 Time Zone Format

(Address: 2379, Total 4096 bytes)

Index	First set		…….		8th set	
1	Time Start HHMM	Time End HHMM				
2						
..						
..						
254						

- Time Start :
  Takes up 2 bytes, first byte contains hour, second byte contains minute.
  Example: 13:25➔ 0x13, 0x25 (BCD format)

- Time End
  Takes up 2 bytes, first byte contains hour, second byte contains minute.
  Example: 17:01➔ 0x17, 0x01 (BCD format)

※ Note: Compares up to 8 time schedules in a day, When 0xFF, 0xFF is placed in time start, it indicates that it will not perform comparison.

2.3.6 Transaction Data and Event Log Format

Device ID (1Byte)+Time (4Byte)+Status (1Byte) +Card No (12 bytes) +Event Code (12 byte)

- Device ID (1Byte)

IF ID = 255, the record is transmitted by the controller, else the record is transmitted by the reader.

- Time (4Byte) byte1, byte2, byte3, byte4

Byte 4	Byte 3	Byte 2	Byte 1
bit 7…bit0	bit7….bit0	bit 7….bit0	bit7….bit0
000000 00	00 00000 0	0000 0000	00 000000

Year Month Date Hour Minute Second

- Condition Options:

Code	Description
0x00	Signifies event and followed by an event code
0x01	Valid card
0x04	Authorization mismatch
0x05	Holiday authorization mismatch
0x06	Card number expired
0x07	Card number does not exist
0x0B	Trigger alarm for blacklist card
0x0C	Deactivate alarm
0x16	Swiped a card during unlock door time schedule
0x18	Open door without card swipe(ex.: Open door via exit button)

- Event Code:

Code	Description
031B01	Trigger alarm for blacklist card (Controller)
031B02	Activaate reswipe card alarm (Controller)
020101	Inspector switch on (Controller)
031C01	Deactivate blacklist card alarm (Controller)
031C02	Deactivate reswipe card alarm (Controller)
020201	Inspector switch off (Controller)
0301	Activate tamper sensor alarm
0305	Deactivate tamper sensor alarm
0306	Cold boot (Controller)
020102	Stop switch on
020202	Stop switch off
0117	Swipe master card

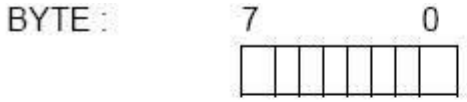
- Card Tyep: Appends 0 if card number less than 12 byte.
- Event Code: Appends 0 if event code less than 12 byte.

Appendix 3: RAC-960/970 series Description

3.1 Basic Format Description

RAC-960/970 ID range: 1-255 (RS-232/485 Communication Type)

System byte indicates:

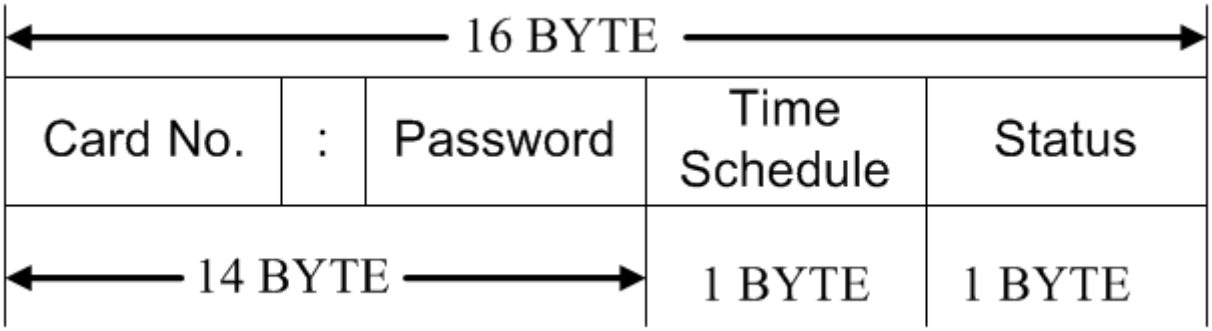


DOOR1:Reader1 ID : 0001
Reader2 ID : 0002

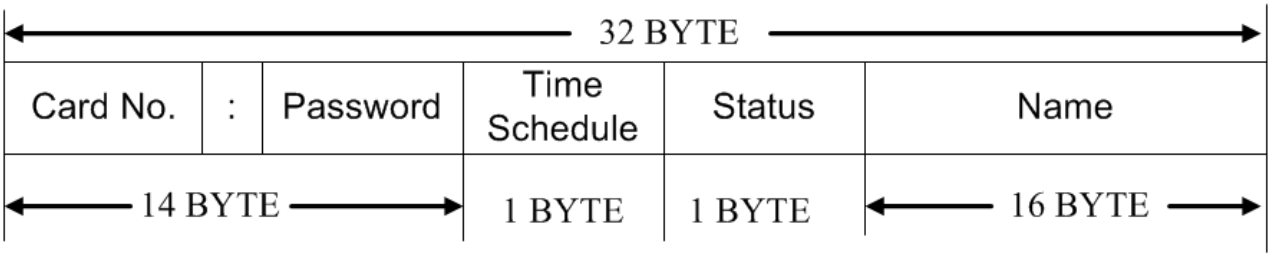
DOOR2: Reader1 ID : 0003
Reader2 ID : 0004

3.2 Valid Card Format

Standard Mode：



Valid Card and Display Name



Kindly refer to 0013H of " Flash Memory Allocation".

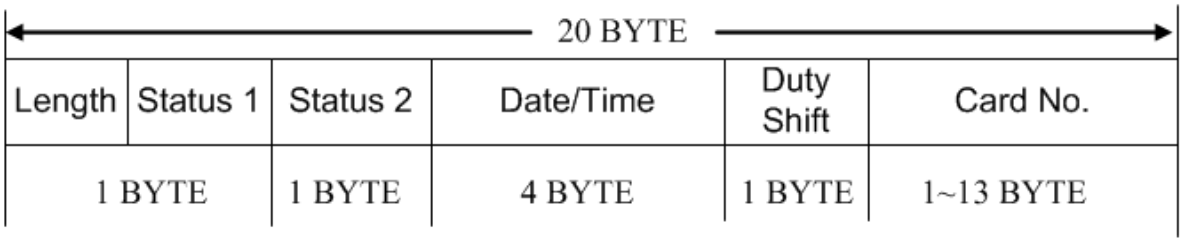
1	0013H	Device Mode 0=Standard Mode (same with RAC-2000) 1=Valide Card & display name	00H	

- Card No. ASCII
- Password: Appends FFH if insufficient or no password is provided.
- The password is compressed by combining 2 digits into 1 byte.
- ":",【Colon】is used to separate the card number from the password.
- Provide 128 time shceuldes. (0~127)
- Status Explanation：

Bit 0	0= Valid Card 1= Blacklist
Bit 1	960/970 Reserved
Bit 2	0= Do not need check holiday 1= Check holiday
Bit 3	0= Do not need check time schedule 1= Check time schedule
Bit 4~5	960/970 Reserved
Bit 6~7	Reserved

- Name：Please appended 20H when name is hid.

3.3 Swiped Card Records



- Length: 4bit indicates the length of the card number
- Status 1:

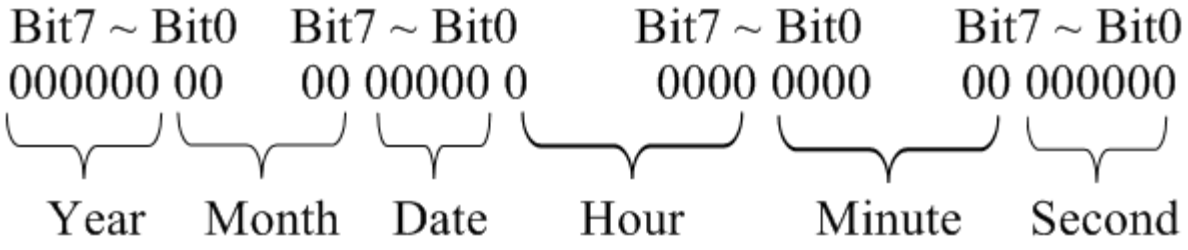
  bit0 0= requires input of card number
  1= swipe card
  bit1~2 0= Inside reader
  1= Outside reader

2= 960/970 Reserved
3= 960/970 Reserved
bit3 0= Swipe card records
1= Event records

- Status 2: bit 0~7 0= Swipe of valid card
  1= Master card
  2= Disarm code
  3= Duress code
  5= Temporary card
  6= Black list
  10= Guest card
  11= Guest card (unlimited usage)
  20= Card not found
  21= Authorization failed (Wrong time schedule)
  61= Swipe of valid card + correct input of password
  62= Swipe of valid card + incorrect input of password
  63= Input of card no. + password
  67= Anti-passback error
  75= Error length of compare valid digits

- Date and time: Total of 4 byte(transmission starts with Lo byte)
  0-5 bit contains seconds
  6-11 bit contains minute
  12-16 contains hours
  17-21 contains date,
  22-25 contains month
  26-31 contains year (2000+N)

- Time(4byte) Byte1,Byte2,Byte3,Byte4



- Duty Shift :

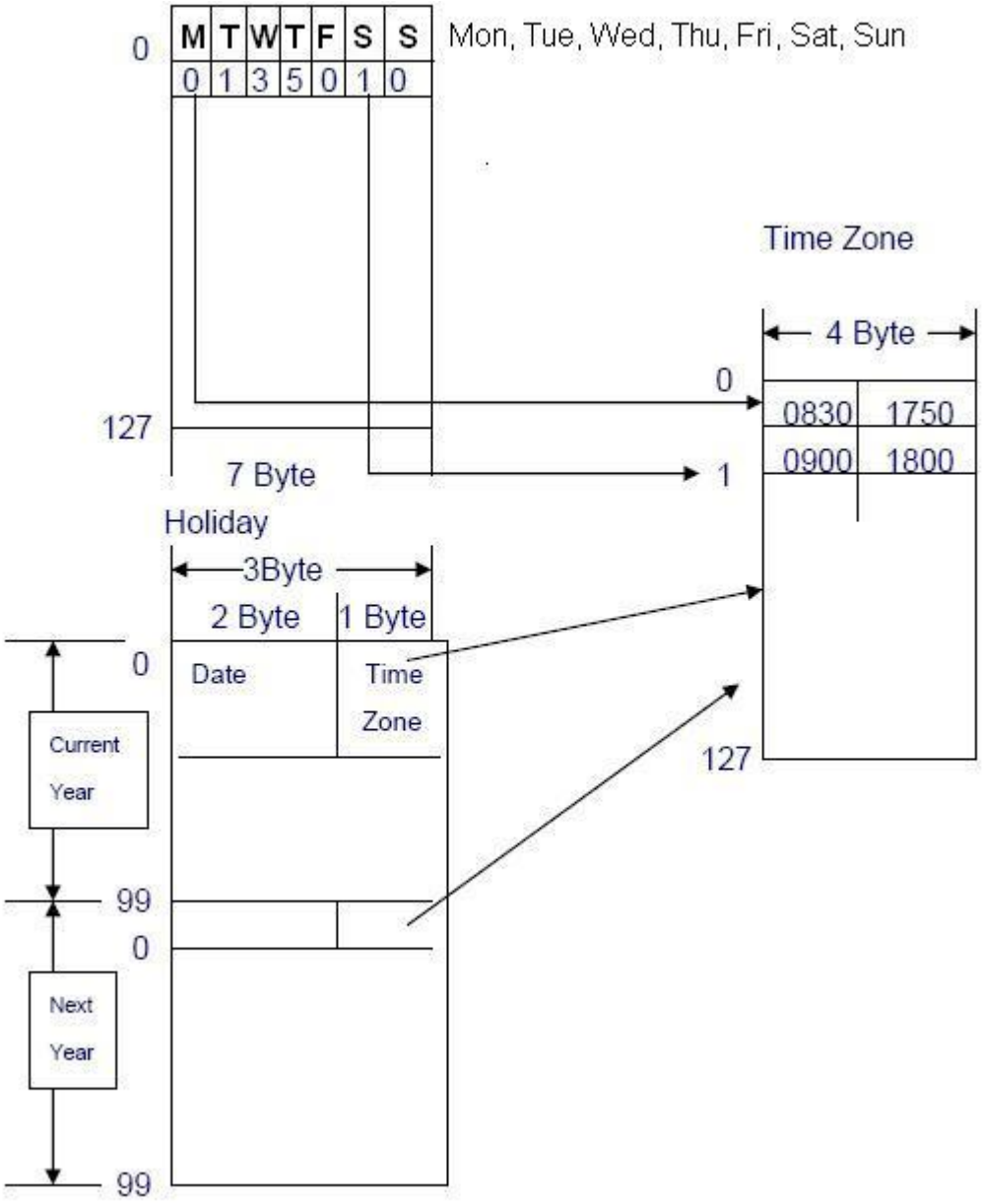
0= BLANK	1=DUTY ON
2=DUTY_OFF	3=BREAK OUT
4=BREAK_IN	5=OT_START
6=OT_END	

- Card number stored is base on length.

＊ When polling with duty shift information, kindly use 1FH command. 10H Command does not support polling with duty shift information.

3.4 Time Schedule

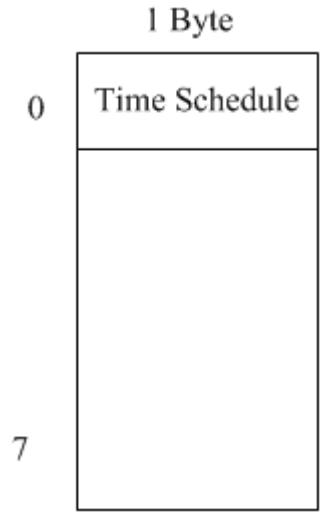
Time Schedule, Time Zone, Holiday



∗ Note: FF is appended if time zone, time schedule and holiday schedule is not provided.

3.5 Unrestricted Time Schedule

Unrestricted Time Schedule 8 sets



3.6 Event Code's Return Value (Sensor / Relay)

0120H	Duress code recognized
0122H	Disarm code deactivates alarm
0123H	Master card(code) modified
0301H	Tamper → Case Sensor
0306H	Cold startup
0307H	Reset
0311H	Door forced open → Door Sensor (Door has been breakthrough)
0312H	Door prop alarm
0313H	Deactivate door prop alarm
0318H	Door opened → Valid access
0319H	Door opened → Door relay closed
031AH	Exit button pressed → Push button
0401H	Memory full
041BH	System initialization

3.7 Flash Memory Allocation

(Address 0000H~00FFH：Total 256Bytes).
Address 0000H~0013H are system parameter.

Length	ADD	Description	Default	Remarks
1	0000H	Communication: Bit7 = 0：RS-485/232 Bit0~6：Baudrate 1=9600 2=19200 3=38400 4=115200 Bit7 = 1：TCP/IP Bit0~6：Reserved	02H	
1	0001H	Reader Format 0=Mifare, Slave reader is T1 interface 1=EM, Slave reader is T1 interface 2= no scanning module, Slave reader is Wiegand interface	00H	
4	0002H	IP		
4	0006H	Mask		
4	000AH	Gateway		
2	000EH	Port		
2	0010H	Max. valid card number (As characteristic of Flash Memory, the valid card no. should be divisible by 256 after modify . Available modify range from 6144~15104)	00H,28H	Lo Byte,Hi Byte 2800H = 10240 set
1	0012H	Controller ID	01H	RS-232/485 used
1	0013H	Controller Mode 0=Standard Mode (same with RAC-2000) 1=Valid Card + Display name	00H	
1	0014H	Controller status setup Bit0：Slave reader doesn’t need check time schedule 0= Do not need check time schedule. 1=Check Time schedule	00H	

		Bit1：Inside/Outside Mode selection 0=Inside 1=Outside Bit2：Activate alarm relay for blacklist. 0=Disable 1=Enable Bit3: Reqeust password for slave reader 0=Disable (Request password) 1=Enable (No need password) Bit4: Display card number 0=Disable (Display * mark) 1=Enable (Display card number) Bit5：Reserve invalid card records or not 0= Yes 1= No Bit6：Overwrite stored data or not 0= Overwirte 1= No, Alarm warning when records reach to 90%. Bit7：Activate alarm relay when memory is full. 0=Disable 1=Enable		
1	0015H	Year of the holiday schedule	09H	
10	0016H	Mifare setup	00H,00H,00H, 00H, FFH,FFH,FFH, FFH,FFH,FFH	Refer to R/W Table 54
1	0020H	Display date format in LCD 0=YYYY/MM/DD 1=MM/DD/YYYY 2=DD/MM/YYYY	00H	
1	0021H	LCD status setup	10H	Default :

		Bit0~3 = Reserved Bit4~5 = LCD backlight mode (0=Auto,1=Enable,2=Disable) Bit6~7 = Reserved		LCD backlight is enabled.
1	0022H	Tiem of LCD back to ready Mode 0= Default is 8 seconds. 1~255=1~255 seconds	00H	
1	0023H	Keypad setup Bit0 = Keypad action (0=Disable,1=Enable) Bit1~3 = Reserved Bit4~5 = Keypad with backlight mode (0=Auto,1=Enable,2=Disable) Bit6~7 = Reserved	01H	Default: Keyad action is enabled and keypad backlight is in auto model.
2	0024H	Retrieve valid digits length of valid card	01H, 00H	
2	0026H	Compare valid digits length of valid card.	00H, 00H	
1	0028H	0=Anti Disable 1=Anti Enable	00H	
1	0029H	Language selection 0 = English 1 = Traditional Chinese 2 = Simplified Chinese	00H	
1	002AH	Sensor status (1=Close,0=Open) Bit0 = 0 (Case Sensor) Bit1 = 0 (Door Sensor) Bit2 = 1 (Push Button) Bit3~7 = Reserved // bit0,1=Case1,2 bit4,6=PushButton1,2 bit5,7=DoorSensor1,2	04H	
1	002BH	Case Sensor will activate alarm Bit0 = Buzzer Bit1 = Alarm Relay activate Bit2~7 = Reserved // bit0,4 = 1 Reader Buzzer buzzer, bit1,5 = 1 Alarm Reader buzzer	01H	
1	002CH	Door Sensor detection time 0= Close door sensor 1~255= Detection time1~255 seconds	00H	

<table>
<tr><td>1</td><td>002DH</td><td>Door Sensor will activate alarm
Bit0 = Buzzer
Bit1 = Alarm Relay activate
Bit2~7 = Reserved</td><td>01H</td><td></td></tr>
<tr><td>1</td><td>002EH</td><td>Max. failure of swipe card
0= Close
1~255= Failure times 1~255.</td><td>00H</td><td></td></tr>
<tr><td>1</td><td>002FH</td><td>The time for system to cease operation upon reaching max. number of failed log in attempts.
0= Close
1~255=1~255 seconds</td><td>00H</td><td></td></tr>
<tr><td>14</td><td>0030H</td><td>Master Card (FFH is appended to unused space)</td><td>‘30191000’</td><td></td></tr>
<tr><td>1</td><td>003EH</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>1</td><td>003FH</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>14</td><td>0040H</td><td>Disarm Code (FFH is appended to unused space)</td><td>‘0000’</td><td></td></tr>
<tr><td>1</td><td>004EH</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>1</td><td>004FH</td><td>Relay mode selection
<table><tr><td></td><td>Relay0</td><td>Relay1</td><td>Relay2</td><td>Relay3</td></tr><tr><td>Mode 0</td><td>Door</td><td>Alarm</td><td>Alarm</td><td>Bell</td></tr><tr><td>Mode 1</td><td>X</td><td>Door</td><td>Alarm</td><td>Bell</td></tr><tr><td>Mode 2</td><td>X</td><td>Door</td><td>Alarm</td><td>Alarm</td></tr></table></td><td>00H</td><td>Relay 0 contains relay of controller. Relay1~3 contains relay of ACU-60.</td></tr>
<tr><td>14</td><td>0050H</td><td>Duress Code (FFH is appended to unused space)</td><td>‘1190’</td><td></td></tr>
<tr><td>1</td><td>005EH</td><td>Repeat activate mode of alarm relay
Bit0~5
0 = Activate alarm relay once
1~63= Inactivate alarm after activate alarm for few minitus.
Bit6~7 = Reserved</td><td>00H</td><td>It will only work in Latch mode of alarm relay.</td></tr>
<tr><td>1</td><td>005FH</td><td>In repeat activate mode, relay activate time and repeat activate frequency.
Bit0~4
1~31= After 1~31 minutes, activate</td><td>00H</td><td>It will only work in Latch mode of alarm</td></tr>
</table>

		alarm relay again. Bit5~7 1~7 = frequency 1~7 .		relay.
2	0060H	Relay0 setup (Door) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 seconds Bit12~14 = Reserved Bit15 = Other mode (Time of Pulse is 0 second.) 0=Toggle Relay 0 does not support Latch mode.	28H,00H	Lo Byte,Hi Byte 0028H=40 40*0.1 秒=4 秒
2	0062H	Relay1 setup (Alarm) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 seconds Bit12~14 = Reserved Bit15 = Other mode (Time of Pulse is 0 second.) 0=Toggle 1=Latch	00H,80H	Lo Byte,Hi Byte 0028H=40 40*0.1 秒=4 秒
2	0064H	Reserved		
2	0066H	Relay3 setup (Bell) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 seconds Bit12~14 = Reserved Bit15 = Other mode (Time of Pulse is 0 second.) Relay3 does not support Toggle and Latch mode.	28H,00H	Lo Byte,Hi Byte 0028H=40 40*0.1 second=4 seconds
	0068H	0068H~00FFH Reserved		

3.7.1 Time Schedule

(Address 0100H~047FH： Total 896Bytes)

128*7	0100H	Time schedule 128 sets*7Byte		

Byte1：Corresponding time schedule of Monday (00H~7FH 128 sets)

Byte2：Corresponding time schedule of Tuesday (00H~7FH 128 sets)

…….

Byte7：Corresponding time schedule of Sunday (00H~7FH 128 sets)

＊ FFH is appended to unused space.

3.7.2 Time Zone

(Address 0480H~067FH：Total 512Bytes)

128*4	0480H	Time Zone 128 sets*4Byte		

Start Time：2 byte, First byte is hour. Second byte is minute.

Example : 13：45 => 13H,45H (BCD format)

End Time：2 byte, First byte is hour. Second byte is minute.

Example : 20：12 => 20H,12H (BCD format)

＊ FFH is appended to unused space.

3.7.3 Holiday Schedule

(Address 0680H~08D7H：Total 600Bytes)

200*3	0680H	Holiday schedule 200 sets*3Byte		Current Year has 0~99 sets. Next year has 100~199 sets.

Holiday Date：2 byte, First byte is month. Second byte is date.

Example : December 25 => 12H,25H (BCD format)

Corresponding time schedule：1 byte

Example: Second Time schedule => 01H (00H~7FH 128 sets)

＊ FFH is appended to unused space.

3.7.4 Unrestricted Time Schedule

(Address 08D8H~08DFH：Total 8Bytes)

8*1	08D8H	8 sets time schedule of continuted open door		

＊ FFH is appended to unused space.

3.7.5 Alarm Time Schedule

(Address 08E0H~097FH：Total 160Bytes)

32*5	08E0H	Alarm time schedule 32 sets*5Byte		

Start Time：2 byte, First byte is hour. Second byte is minute.

Example : 08：30 => 08H,30H (BCD format)

Action Time：2 byte, First byte is minute. Second byte is second.

Example : 00：12 => 00H,12H (BCD format)

Weekday：1 byte

bit7： Reserved, bit6~0： Sun/Sat/Fri/Thu/Wed/Tue/Mon

＊ FFH is appended to unused space.

3.7.6 Anti Reset

(Address 0980H~099FH：Total 32Bytes)

16*2	0980H	Time clear Anti sign,16 sets* 2byte		

Time：2 byte, First byteis hour. Second byte is minute.

Example 23：50 => 23H,50H (BCD format)

＊ FFH is appended to unused space.

3.7.7 Duty Shift

(Address 09A0H~09FFH：Total 96Bytes)

32*3	09A0H	32 sets* 3Byte		

Group	Time (2byte)	Duty Shift (1byte)
First group, address 0090H	HH：MM	Class
2th group, address 0093H	HH：MM	Class

3rd group, address 0096H	HH：MM	Class
.	.	.
.	.	.
.	.	.
32 group, address 00EDH	HH：MM	Class

Time：2 byte, First byteis hour. Second byte is minute.

Example : 08：10 => 08H,10H (BCD format)

Duty Shift：1 byte

0=BLANK	1=DUTY ON
2=DUTY_OFF	3=BREAK OUT
4=BREAK_IN	5=OT_START
6=OT_END	

＊ FFH is appended to unused space.

3.7.8 Display Messange

(Address 0A00H~0A7FH：Total 128Bytes)

8*16	0A00H	Message 8 sets* 16Byte		

Group	Message	Length of display	Description
0	HUNDURE	10	Ready Mode
1	DUTY ON	10	Duty On
2	DUTY OFF	10	Duty Off
3	BREAK OUT	10	Break Out
4	BREAK IN	10	Break In
5	OT_START	10	Start Overtime
6	OT_END	10	End Overtime
7		10	Blank

Every message display 10 byte currently.

＊ FFH is appended to unused space.

3.7.9 Reserved

(Address 0A80H~1FFFH)

	0A80H	0A80H-1FFFH BLANK		

3.7.10 Valide Card

(Address 002000H~027FFFH)

(10240 pcs * 16Byte) / 4KB, around 40 Sector.

3.7.11 Swiped Records

(Address 028000H~07FFFFH)

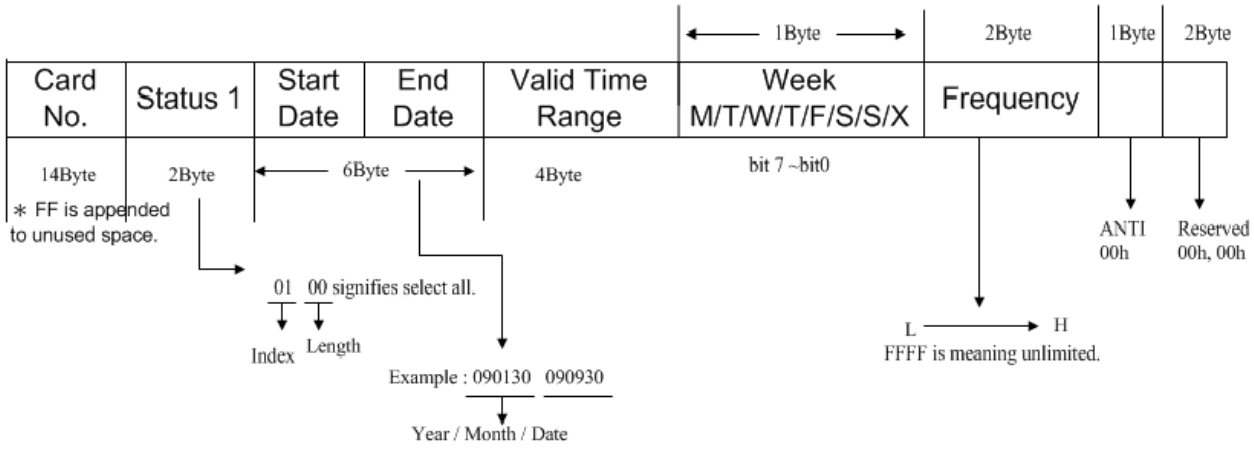
SRAM Memory Plan (R/W Table 15,16)

3.7.12 SRAM Memory Allcation

(R/W table 15, 16)

◎ Guest Card: 50 sets (Address 0000H ~ 063FH : Total 1600Bytes)

50*32	0000H	50 sets Guest Card * 32byte		



3.7.13 Event Records 200 sets

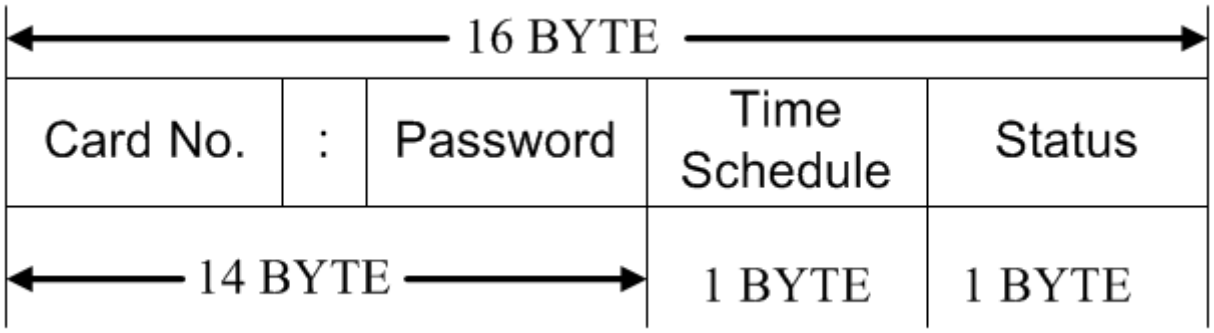
(Address0640H~0C7FH：Total 1600Bytes)

200*8	0640H	Event records 200 sets* 8Byte		

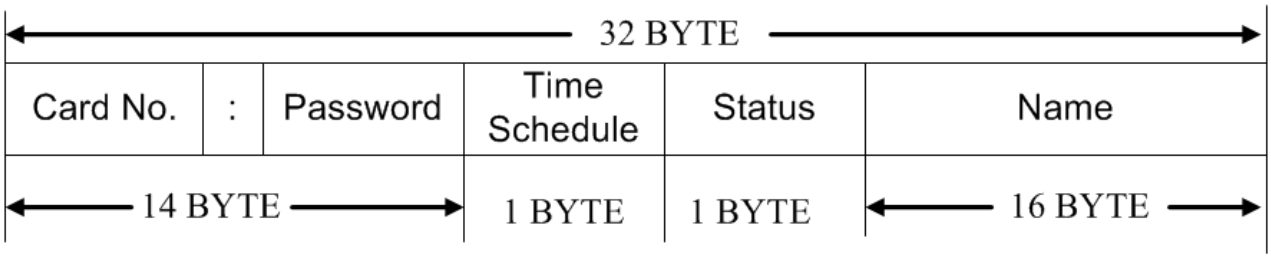
Appendix 4 : RAC-940 series Description

4.1 Valid Card Format

Standard Mode：



Valid Card and Display Name



- Card No. and Passowrd : Total is 14 byte.
- Card number is ASCII
- The password is compressed by combining 2 digits into 1 bytd. Appends FFH if insufficient or no password is provided.
- ":",【Colon】is used to separate the card number from the password.

- Time Schedule : 0~15 index to time schedule 1.
  128~143 index to time schedule 2.

  1.The different between time schedule 1 and 2 is time schedule 1 only corresponding a set of time zone. Time schedule 2 corresponding max. 8 sets of time zone.

  2.It is hightly recommends to use time schedule 2 when need multi-time zones a day for access. It provides high efficiency and do not occupy valid card space.

- Status Explanation：

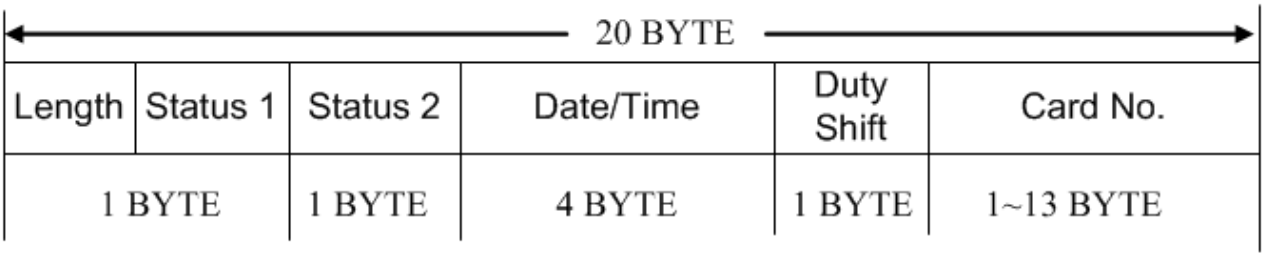
Bit 0	0= Valid Card

	1= Blacklist
Bit 1	Reserved
Bit 2	0= Do not need check holiday 1= Check holiday
Bit 3	0= Do not need check time schedule 1= Check time schedule
Bit 4~5	Reserved
Bit 6~7	Reserved

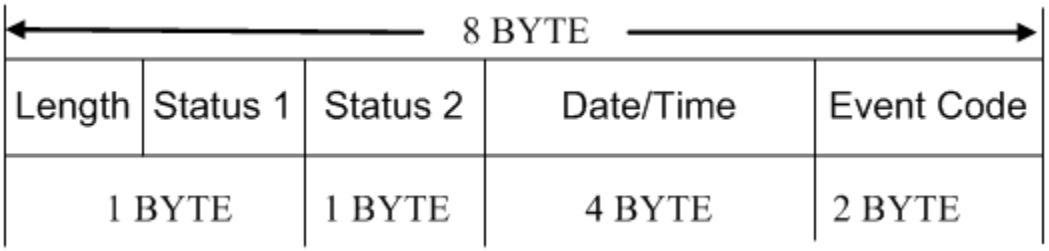
- Name：Please appended 20H when name is hid.

4.2 Swiped Card Records and Event Format

◎ **Swiped Card Format**



◎ **Event Format**



- Length: Bit7~4 Indicates the length of the card number
- Status 1:

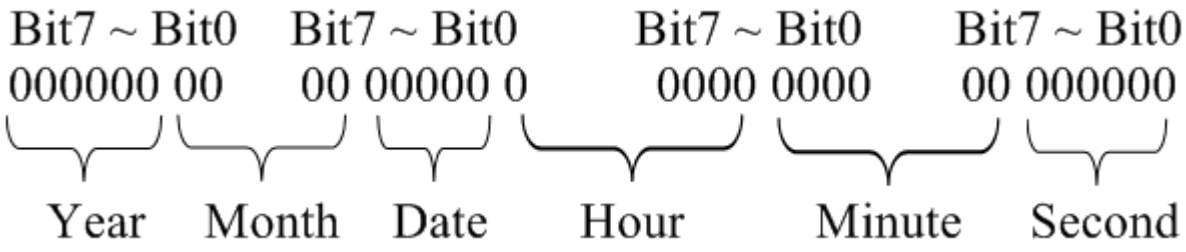
bit0 0= requires input of card number
1= swipe card
Bit1 0= Inside reader
1= Outside reader
Bit2 Reserved
Bit3 0= Swipe card records

1= Event records

- Status 2: Bit 0~7 0= Swipe of valid card
  1= Master card
  2= Disarm code
  3= Duress code
  6= Black list
  20= Can not found
  21= Authorization failed (Wrong time schedule)
  61= Swipe of valid card + correct input of password
  62= Swipe of valid card + incorrect input of password
  63= Input of card no. + password
  67= Anti-passback error
  75= Error length of compare valid digits

- Date and time: Total of 4 byte(transmission starts with Lo byte)
  0-5 bit contains seconds
  6-11 bit contains minute
  12-16 contains hours
  17-21 contains date,
  22-25 contains month
  26-31 contains year (2000+N)

- Time(4byte) Byte1,Byte2,Byte3,Byte4



- Duty Shift :

0= BLANK	1=DUTY ON
2=DUTY_OFF	3=BREAK OUT
4=BREAK_IN	5=OT_START

6=OT_END	

- Event Code's Return Value (Sensor / Relay)

0120H	Duress code recognized
0122H	Disarm code deactivates alarm
0123H	Master card(code) modified
0301H	Tamper → Case Sensor
0306H	Cold startup
0307H	Reset
0311H	Door forced open → Door Sensor (Door has been breakthrough)
0312H	Door prop alarm
0313H	Deactivate door prop alarm
0318H	Door opened → Valid access
0319H	Door opened → Door relay closed
031AH	Exit button pressed → Push button
0401	Memory full
041BH	System initialization

4.3. Flash Memory Allocation

(Address 0000H~0013H：Total 20Bytes)

Length	ADD	Purpose	Default	Remarks
1	0000H	Communication : Bit0~6：Baudrate 1=9600 2=19200 3=38400 4=115200 Bit7：Reserved	02H	
1	0001H	Reserved		
4	0002H	Reserved		
4	0006H	Reserved		

4	000AH	Reserved		
2	000EH	Reserved		
2	0010H	Max. Valid card number, include blacklist, Max. are 5000 sets.	88H,13H	Lo Byte,Hi Byte 1388H = 5000 sets
1	0012H	Controller ID	01H	
1	0013H	Controller Mode 0= Standard Mode (Same with RAC-2000) 1=Valid Card + Display name	00H	

4.4 General Parameter Allocation

(Address 0014H~00FFH：Total 236Bytes)

Length	ADD	Purpose	Defautl	Remark
1	0014H	Controller status setup Bit0：Slave reader doesn't need check time schedule 0= Do not need check time schedule. 1=Check Time schedule Bit1：Inside/Outside Mode selection 0=Inside 1=Outside Bit2：Activate alarm relay for blacklist. 0=Disable 1=Enable Bit3~4：Reserved Bit5：Reserve invalid card records or not 0= Yes 1= No Bit6：Overwrite stored data or not 0= Overwrite 1= No, Alarm warning when records reach to 90%.	00H	

		Bit7：Activate alarm relay when memory is full. 0=Disable 1=Enable		
1	0015H	Year of the holiday schedule	09H	
10	0016H	Reserved		
1	0020H	Display date format in LCD 0=YYYY/MM/DD 1=MM/DD/YYYY 2=DD/MM/YYYY	00H	
1	0021H	LCD status setup Bit0~3 = Reserved Bit4~5 = LCD backlight mode (0=Auto,1=Enable,2=Disable) Bit6~7 = Reserved	10H	Default : LCD backlight is enabled.
1	0022H	Time of LCD back to ready Mode 0= Default is 8 seconds. 1~255=1~255 seconds	00H	
1	0023H	Keypad setup Bit0 = Keypad action (0=Disable,1=Enable) Bit1~3 = Reserved Bit4~5 = Keypad with backlight mode (0=Auto,1=Enable,2=Disable) Bit6~7 = Reserved	01H	Default: Keypad action is enabled and keypad backlight is in auto model.
2	0024H	Retrieve valid digits length of valid card	01H, 00H	
2	0026H	Compare valid digits length of valid card.	00H, 00H	
1	0028H	0=Anti Disable 1=Anti Enable	00H	
1	0029H	Language selection 0 = English 1 = Traditional Chinese 2 = Simplified Chinese	00H	
1	002AH	Sensor status (1=Close,0=Open) Bit0 = Reserved Bit1 = 0 (Door Sensor) Bit2 = 1 (Push Button) Bit3~7 = Reserved // bit0,1=Case1,2 bit4,6=PushButton1,2	04H	

<table>
<tr><td></td><td></td><td>bit5,7=DoorSensor1,2</td><td></td><td></td></tr>
<tr><td>1</td><td>002BH</td><td>Case Sensor will activate alarm
Bit0 = Buzzer
Bit1 = Alarm Relay activate
Bit2~7 = Reserved
// bit0,4 = 1 Reader Buzzer buzzed,
bit1,5 = 1 Alarm Reader buzzer</td><td>01H</td><td></td></tr>
<tr><td>1</td><td>002CH</td><td>Door Sensor detection time
0= Close door sensor
1~255= Detection time1~255 seconds</td><td>00H</td><td></td></tr>
<tr><td>1</td><td>002DH</td><td>Door Sensor will activate alarm
Bit0 = Buzzer
Bit1 = Alarm Relay activate
Bit2~7 = Reserved</td><td>01H</td><td></td></tr>
<tr><td>1</td><td>002EH</td><td>Max. failure of swipe card
0= Close
1~255= Failure times 1~255.</td><td>00H</td><td></td></tr>
<tr><td>1</td><td>002FH</td><td>The time for system to cease operation upon reaching max. numbers of failed log in attempts.
0= Close
1~255=1~255 seconds</td><td>00H</td><td></td></tr>
<tr><td>14</td><td>0030H</td><td>Master Card
(FFH is appended to unused space)</td><td>‘30191000’</td><td></td></tr>
<tr><td>1</td><td>003EH</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>1</td><td>003FH</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>14</td><td>0040H</td><td>Disarm Code
(FFH is appended to unused space)</td><td>‘0000’</td><td></td></tr>
<tr><td>1</td><td>004EH</td><td>Reserved</td><td></td><td></td></tr>
<tr><td>1</td><td>004FH</td><td>Relay mode selection
<table><tr><td></td><td>Relay0</td><td>Relay1</td><td>Relay2</td><td>Relay3</td></tr><tr><td>Mode 0</td><td>Door</td><td>Alarm</td><td>Alarm</td><td>Bell</td></tr><tr><td>Mode 1</td><td>X</td><td>Door</td><td>Alarm</td><td>Bell</td></tr><tr><td>Mode 2</td><td>X</td><td>Door</td><td>Alarm</td><td>Alarm</td></tr></table></td><td>00H</td><td>Relay 0 contains relay of controller. Relay1~3 contains relay of ACU-60.</td></tr>
<tr><td>14</td><td>0050H</td><td>Duress Code
(FFH is appended to unused space)</td><td>‘1190’</td><td></td></tr>
</table>

1	005EH	Repeat activate mode of alarm relay Bit0~5 0 = Activate alarm relay once 1~63= Inactivate alarm after activate alarm for few minutes. Bit6~7 = Reserved	00H	It will only work in Latch mode of alarm relay.
1	005FH	In repeat activate mode, relay activate time and repeat activate frequency. Bit0~4 1~31= After 1~31 minutes, activate alarm relay again. Bit5~7 1~7 = frequency 1~7 .	00H	It will only work in Latch mode of alarm relay.
2	0060H	Relay0 setup (Door) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 seconds Bit12~14 = Reserved Bit15 = Other mode (Time of Pulse is 0 second.) 0=Toggle Relay 0 does not support Latch mode.	28H,00H	Lo Byte,Hi Byte 0028H=40 40*0.1 second=4 seconds
2	0062H	Relay1 setup (Alarm) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 seconds Bit12~14 = Reserved Bit15 = Other mode (Time of Pulse is 0 second.) 0=Toggle 1=Latch	00H,80H	Lo Byte,Hi Byte 0028H=40 40*0.1 second=4 seconds
2	0064H	Reserved		
2	0066H	Relay3 setup (Bell) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 seconds Bit12~14 = Reserved Bit15 = Other mode (Time of Pulse is 0 second.) Relay3 does not support Toggle and	28H,00H	Lo Byte,Hi Byte 0028H=40 40*0.1 second=4 seconds

		Latch mode.		
1	0068H	Max. number of blacklist 0~255	64H	
	0069H	0069H~00FFH Reserved		

4.5 Time Schedule 1

(Address : 0100~016F : Total 112 Bytes. 16 sets)

Index	Mon.	Tue.	Med.	Thu.	Fri.	Sat.	Sun.
0	Time Schedule Index						
1							
..							
..							
15							

Weekend

● Time Schedule index: 0 ~15 → references to the set time schedule

255 → signifies no time shchedule set

4.6 Time Zone 1

(Address : 0480H~04BFH : Total 64 Bytes. 16 sets)

Index	2 byte	2 byte
0	Time Start HH:MM	Time End HH:MM
1		
..		
..		
15		

● Time Start :

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 13:25➔ 13H, 25H (BCD format)

● Time End

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 17:01➔ 17H, 01H (BCD format)

4.7 Holiday Schedule

(Address : 0680H~07FFH : Total 384 Bytes. 64 *2 sets)

	First Holiday (3 byte)			.....			64th Holiday		
Current Year	Month	Date	Time Schedule Index						
Next Year									

● Time Schedule index: 0~15 → references to the set time schedule1
128~143 → references to the set time schedule2

Note:

1. Month and date are in BCD format, example: 12/1 ➔ 12H, 01H
2. When you place FFH,FFH in month and date, it signifies no comparison of authorization needed.

4.8 Unrestricted Time Schedule

(Address : 08D8H~08DFH : Total 8 Bytes. 8 sets)

1	2	3	4	5	6	7	8
Time Schedule Index							

● Time Schedule index: 0~15 → references to the set time schedule1
128~143 → references to the set time schedule2
255 → signifies the end

4.9 Siren Timetable

(Address : 08E0H~0907H : Total 40 Bytes. 8 sets)

2 Byte	2 Byte	1 Byte
Time Start HHMM	Action Time MMSS	Action Week

0			
1			
.			
.	.		
.	.		
7			

● Time Start :

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 08:30➔ 08H, 30H (BCD format)

● Action Time

Takes up 2 bytes, first byte contains minute, second byte contains second.

Example: 00:12➔ 00H, 12H (BCD format)

● Action Week

1 Byte

bit7 : Reserved

bit6~0 : Sun. Sat. Fri. Thu. Wed. Tue. Mon.

※ FFH is appended when the end.

4.10 Anti Reset Timetable

(Address : 0980H~098FH : Total 6 Bytes. 8 sets)

	2 Byte
0	Activate Time HHMM
1	
.	.
.	.
6	
7	

● Activate Time :

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 23:50➔ 23H, 50H (BCD format)

※ FFH is appended when the end.

4.11 Display Message

(Address : 0A00H~0A7FH : Total 128 Bytes. 8 sets)

	16 Byte
0	Display Message
1	
.	.
.	.
6	
7	

Group	Message	Length of display	Description
0	HUNDURE	10	Ready Mode
1	DUTY ON	10	Duty On
2	DUTY OFF	10	Duty Off
3	BREAK OUT	10	Break Out
4	BREAK IN	10	Break In
5	OT_START	10	Start Overtime
6	OT_END	10	End Overtime
7		10	Blank

Every message display 10 byte currently.

∗ 20H is appended to unused space.

4.12 Request Password Time Schedule

(Address : 0B00H~0B07FH : Total 8 Bytes. 8 sets)

1	2	3	4	5	6	7	8
Time Schedule Index							

● Time Schedule index: 0~15 → references to the set time schedule1

128~143 → references to the set time schedule2

255 → signifies the end

4.13 Alarm Time Schedule

(Address : 0C00H~0C03H : Total 8 Bytes. 8 sets)

2 byte	2 byte
Time Start HHMM	Time End HHMM

● Time Start :

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 13:25➔ 13H, 25H (BCD format)

● Time End

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 17:01➔ 17H, 01H (BCD format)

4.14 Time Schedule 2

(Address : 0C04~0C73 : Total 112 Bytes. 16 sets)

Weekend

	Mon.	Tue.	Med.	Thu.	Fri.	Sat.	Sun.
128	Time Schedule Index						
129							
..							
..							
143							

● Time Schedule index: 0 ~15 → references to the set time zone 2

255 → signifies no time shchedule set

4.15 Time Zone 2

(Address : 0CE4~0EE3 : Total 512 Bytes. 16 sets)

Index	Group 1		…….		Group 8	
0	Time Start HH:MM	Time End HH:MM				
2						
..						
..						
15						

● Time Start :

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 13:25➔ 13H, 25H (BCD format)

● Time End

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 17:01➔ 17H, 01H (BCD format)

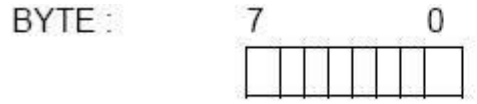
Note: Compares up to 8 time schedules in a day, When 0xFF, 0xFF is placed in time start, it indicates the end.

Appendix 5: RAC-2200 series Description

5.1 Basic Format Description

RAC-2200 ID range : 1 – 255

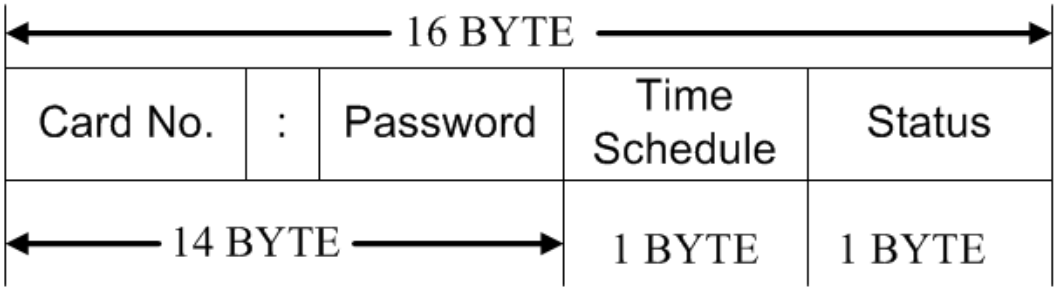
System byte indicates:



Reader ID Setup

	Outside Reader ID	Inside Readaer ID
Door 1	0	16
Door 2	1	17
Door 3	2	18
Door 4	3	19

5.2 Valid Card Format



- Status：

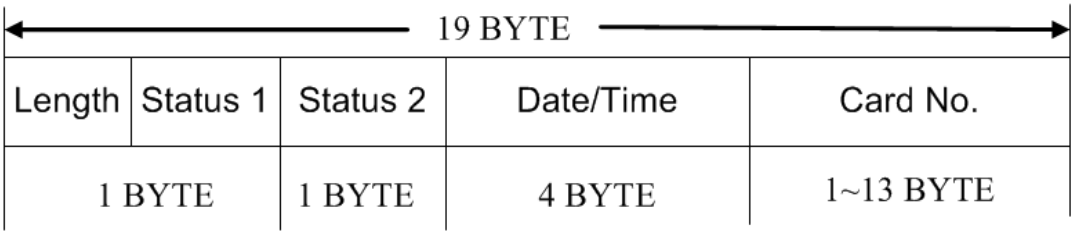
bit0	1= Blacklist
biy1	Reserved
bit2	0= Do not need check holiday 1= Check holiday
bit3	0= Do not need check time schedule 1= Check time schedule
bit4	0= Initial Value of Anti started 1= This card is existent.

bit5	0= Last swipe card is IN 1= Last swipe card is OUT
bit6~7	0= First door 1= Second door 2= Thrid door 3= Furth door

- Password: Appends FF if insufficient or no password is provided.
- The password is compressed by combining 2 digits into 1 byte.
- ":",【Colon】is used to separate the card number from the password.

5.3 Swiped Card Records

Swiped Card Format



- Length: 4bit Indicates the length of the card number
- Status 1: 4 bit

bit0~2 000 Door 1 Outside readaer
001 Door 1 Inside reader
010 Door 2 Outside readaer
011 Door 2 Inside reader
100 Door 3 Outside readaer
101 Door 3 Inside reader
110 Door 4 Outside readaer
111 Door 4 Inside reader

Bit3 =0 Invalid card number
= 1 Event (In 7th and 8th byte). Data saving is base on length of the card number.

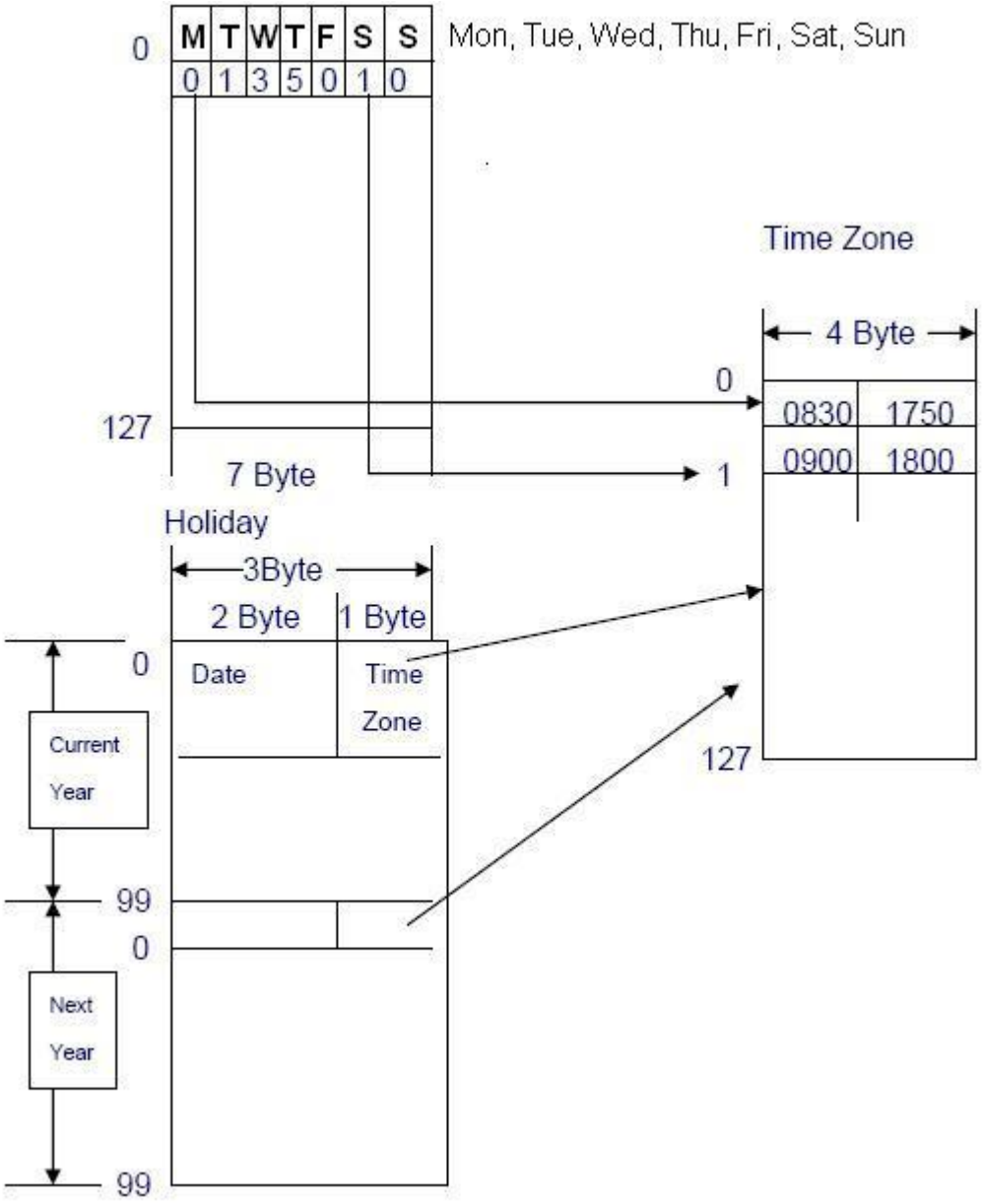
- Status 2: 0= Swipe of valid card
1= Master card
2= Disarm code
3= Duress code
5= Temporary card

6= Blacklist
10= Guest card
11= Guest card (Unlimited usage)
20= Card not found
21= Authorization failed (Wrong time schedule)
61= Swipe of valid card + correct input of password
62= Swipe of valid card + incorrect input of password
63= Input of card no. + password
67= Anti-passback error

- Date and time: Total of 4 byte(transmission starts with Lo byte)
  0-5 bit contains seconds
  6-11 bit contains minute
  12-16 contains hours
  17-21 contains date,
  22-25 contains month
  26-31 contains year (2000+N)

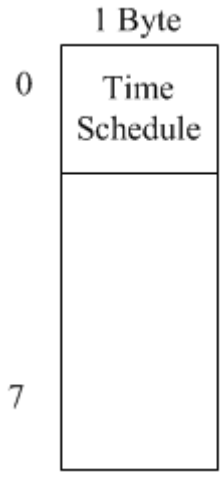
5.4 Time Schedule, Time Zone, Holiday

Time Schedule:

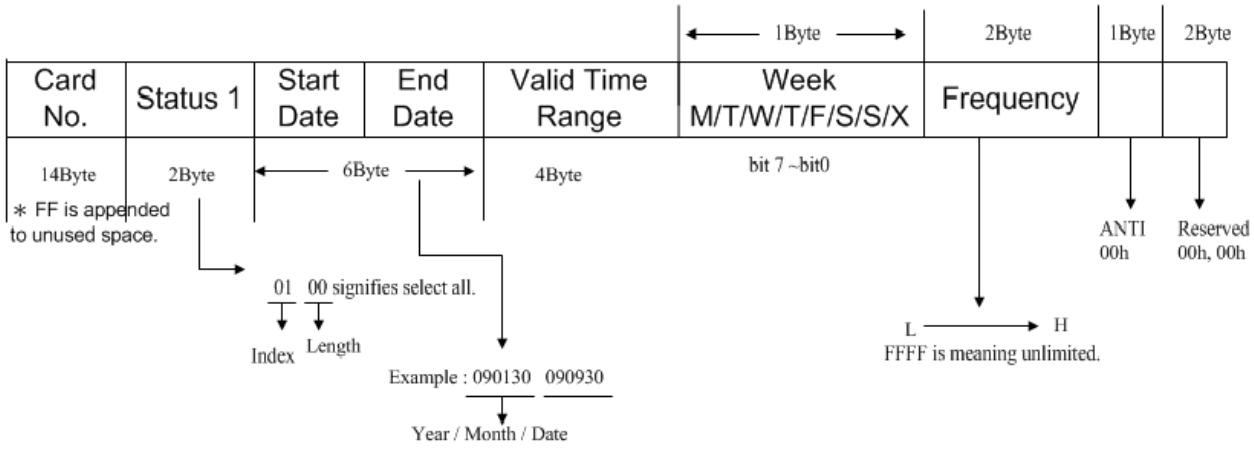


Note: FF is appended if time zone, time schedule and holiday schedule is not provided.

Conditional unlock door schedule : 8 sets



Guest Card : 50 sets. (50 * 32Bytes)



Frequency: Number of store in card of swiped.
(FFFFH is unlimited usage) (Hi, Lo)

5.5 Anti-passback settings

Kindly input 00h when download this byte.

AntiMode = 1 By Door	AntiMode = 2 By Reader
Acivate Door1 Relay IN/OUT	Door1 Relay IN/OUT
Reader ID 0/16/2/18 for IN,	Door1 Relay
Reader ID 1/17/3/19 for OUT	Reader ID 0 for IN,Reader ID 16 for OUT
	Door2 Relay
AntiMode = 3 By Door	Reader ID 1 for IN,Reader ID 17 for OUT
Door1 Relay IN,	Door3 Relay
Door2 Relay OUT	Reader ID 2 for IN,Reader ID 18 for OUT
Reader ID 0/16/2/18 for IN	Door4 Relay

Reader ID 1/17/3/19 for OUT	Reader ID 3 for IN,Reader ID 19 for OUT
bit4： 0：Initial Value of Anti started 1：This card is existent.	bit4~7： 0：Initial Value of Anti started 1：This card is existent. bit4：Door1 bit5：Door2 bit6：Door3 bit7：Door4
bit5： 0：Last swipe is IN 1：Last swipe is OUT	bit0~3： 0：Last swipe card is IN 1：Last swipe card is OUT bit0：Door1 bit1：Door2 bit2：Door3 bit3：Door4

5.6 Event Code's Return Value (Sensor / Relay)

0120H	Duress code recognized
0122H	Disarm code deactivates alarm
0123H	Master card(code) modified
0306H	Cold startup
0307H	Reset
0401H	Memory full
041BH	System initialization

5.7 EEPROM Memory Allocation

Length	ADD	Purpose	Default	Remarks
2	0000H	EEPROM test	5AH,A5H	Startup Test
1	0002H	Device ID	1	
1	0003H	Year of the holiday schedule	08H	
14	0004H	Master card (FFH is appended to unused space)	'30191000'	
14	0012H	Duress card (FFH is appended to unused space)	1190	
14	0020H	Disram code (FFH is appended to unused space)	0000	
1	002EH	Error times of setup	0	
1	002FH	The time for the system to cease operation upon reaching max. no. of failed log in attempts.	0	
2	0030H	The number of valid cards in SRAM	0	
2	0032H	Max. valid card number	10000	
1	0034H	Door1 action time	04H	
1	0035H	Door2 action time	04H	
1	0036H	Door3 action time	04H	
1	0037H	Door4 action time	04H	
1	0038H	Door1 Door Sensor detection time 0= Disable 1~255=1~255 seconds	00H	
1	0039H	Door2 Door Sensor detection time 0= Disable 1~255=1~255 seconds	00H	
1	003AH	Door3 Door Sensor detection time 0= Disable 1~255=1~255 seconds	00H	
1	003BH	Door4 Door Sensor detection time 0= Disable 1~255=1~255 seconds	00H	
1	003CH	Door1 re-swiped card checking time 0= Disable 1~255=1~255 seconds	00H	
1	003DH	Door2 re-swiped card checking time 0= Disable	00H	

		1~255=1~255 seconds		
1	003EH	Door3 re-swiped card checking time 0= Disable 1~255=1~255 seconds	00H	
1	003FH	Door4 re-swiped card checking time 0= Disable 1~255=1~255 seconds	00H	
8	0040H	8 sets conditional unlock time schedule of Door 1	FFH	
8	0048H	8 sets conditional unlock time schedule of Door 2	FFH	
8	0050H	8 sets conditional unlock time schedule of Door 3	FFH	
8	0058H	8 sets conditional unlock time schedule of Door 4	FFH	
1	0060H	**0= NoAnti** **1=Anti By Door** Only Activate Door1 Relay IN/OUT Reader ID 0/16/2/18 for IN, Reader1/17/3/19 for Out **2=Anti By Reader** Door1 Relay IN/OUT Reader ID 0 for IN, Reader ID 16 for OUT Door2 Relay IN/OUT Reader ID 1 for IN, Reader ID 17 for OUT Door3 Relay IN/OUT Reader ID 2 for IN, Reader ID 18 for OUT Door4 Relay IN/OUT Reader ID 3 for IN, Reader ID 19 for OUT **3=Anti By Door** Door1 Relay IN, Door2 Relay OUT Reader ID 0/16/2/18 for IN,	0	

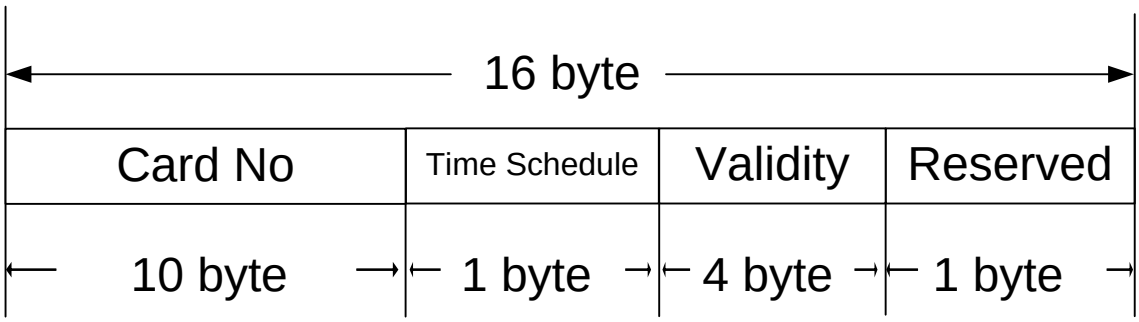
		Reader ID 1/17/3/19 for Out		
1	0061H	Activate relay for black list bit0：0= Disable ,1= Enable	01H	
1	0062H	bit0=1 Memory 90% full alarm bit1=1 Don't overwrite saved records	0	
1	0063H	Blank		
1	0064H	Door1 Reader Keypad status 0= Disable 1= Enable	01H	
1	0065H	Door2 Reader Keypad status 0= Disable 1= Enable	01H	
1	0066H	Door3 Reader Keypad status 0= Disable 1= Enable	01H	
1	0067H	Door4 Reader Keypad status 0= Disable 1= Enable	01H	
152	0068H	Null		
256*7	0100H	Time Schedule 128 sets*7Byte		
256*4	0800H	Time Zone 128 sets*4Byte		
100*3	0C00H	Current Holiday Schedule 100 sets*3Byte		
100*3	0D2CH	Next Year Holiday Schedule 100 sets*3Byte		
	0E58H	0E58H-1FFFH Null		

5.8 SRAM Memory Allocation

Address	Length	Purpose
000000H-00001FH	32	Cache index
000020H-00065FH	1600	Visitor card (50 records*32 byte)
000660H-000C9FH	1600	Saves event (200 records*8 byte)
000CA0H-027D9FH	10000	Valid cards (10000 records*16 byte)
027DA0H-03FFFFH		Swipe card records

Appendix 6: ECU-680 series Description

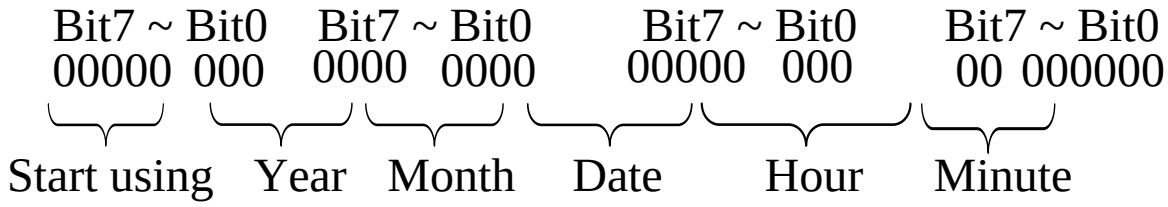
6.1 Valid Card Format



1. Card Number: (10 byte)
   Card number is ASCII code, appends 00H if insufficient.

2. Time Schedule: (1 byte)
   - Bit 0~2: 0~7 Time schedule index
   - Bit 3: 0→ Time schedule controled
     1→ Do not control
   - Bit 4: 0→ Adding card by software
     1→ Adding card by Management Card
   - Bit 5~7: Reserved

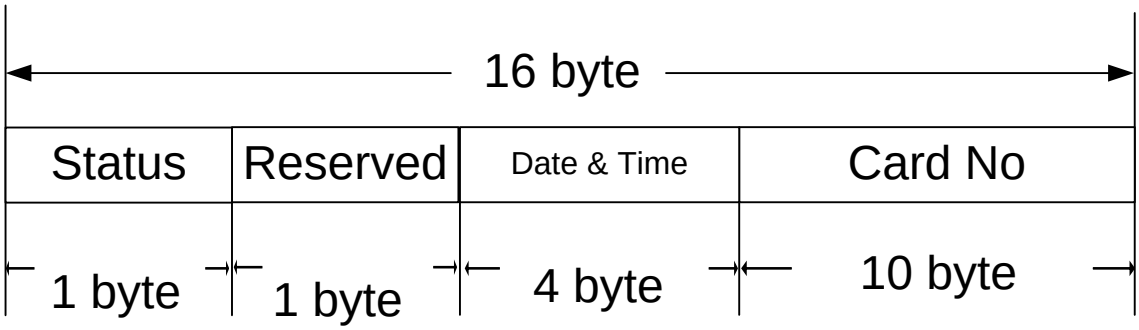
3. Validity: Byte 1,2,3,4 (4 byte)



Note: 00000 signifies activate validity of card. Other values signifies does not use validity.

4. Reserved: (1 byte) Reserved.

6.2 Swiped Card Records Format

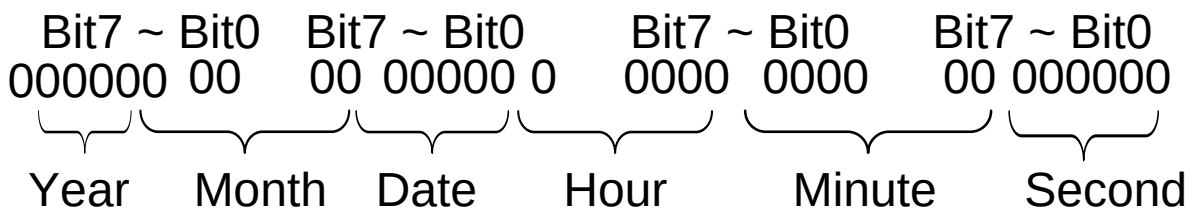


Status:

Bit 0: 0→ Draw out Card
1→ Insert Card
Bit 1: 0→ Invalid Card
1→ Valid Card
Bit 2~6: Reserved
Bit 7: 0→
1→

Date and time: Total of 4 byte(transmission starts with Lo byte)

0-5 bit contains seconds
6-11 bit contains minute
12-16 contains hours
17-21 contains date,
22-25 contains month
26-31 contains year (2000+N)
Time (4bytes) Byte1, Byte2, Byte.3, Byte4



6.3 Flash memory Allocation

(Address 0000H~001FH, Total 32 Bytes)

Length	Address	Description	Default	Remark
2	0000H	Extend Power Off Time	0F, 00H	Second
1	0002H	Power Off Warning Time	00H	Second
1	0003H	Mode: Bit0= Reading sound of swipe card 0→ Disable 1→ Enable Bit1= Store invalid card events 0→ Do not store 1→ Save the events Bit2= Sensor 1 0→ Disable 1→ Enable Bit3= Sensor 2 0→ Disable 1→ Enable Bit4~7= Reserved	00	
2	0004H	Extended Relay Function: 0000H→sample with relay 0001H~FFFFH→ Pulse mode (0.01 ~655.35 seconds)	90H, 01H	0.01 second
1	0006H	Second 1 duration	0	Second
1	0007H	Second 2 duration	0	Second
4	0008H	Mifare module parameter	00H, 1AH, 00H, 01H	
20	000CH	Reserved	FF~	

6.4 Time Schedule

(Address 0020~0057H, Total 56 Bytes, 8 sets)

Index	Weekend Mon.	Tue.	Med.	Thu.	Fri.	Sat.	Sun.
0	Time Schedule Index						
1							
2							
..							
..							
7							

Time Schedule Index: 0~7 index of time schedule

255 is reserved.

6.5 Time Zone

(Address 0058H~00D7H, Total 128 Bytes, 8 sets)

	Group 1		…….		Group 4	
0	Time Start HH:MM	Time End HH:MM				
1						
..						
..						
7						

Time Start :

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 13:25➔ 0x13, 0x25 (BCD format)

- Time End

Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 17:01➔ 0x17, 0x01 (BCD format)

※ Note: Compares up to 4 time schedules in a day, When 0xFF, 0xFF is placed in time start, it indicates that it will not perform comparison.

6.6 Unrestricted Time Schedule

(Address 00D8H~00DBH: Total 4 Bytes, 4 sets)

1	2	3	4
Time Schedule Index			

Time Schedule Index:

0~7 Index with time schedule

255 End

6.7 Management Card

(Address 00DCH~0117H: Total 60 Bytes)

0	Master Card (10 byte)
1	Management Card (10 byte)
2	Management Card (10 byte)
3	Management Card (10 byte)
4	Management Card (10 byte)
5	Management Card (10 byte)

A Master card may set 5 pcs of Management card. Fill 0 if delete the card.

Appendix 7: RAC-820 PEF/RAC-820PMF Format Description

7：RAC-820PEF/RAC-820PMF Format Description

7.1 Valid Card Format

Card No.	:	Password	Time Schedule	Status	Name	Reserved
←	14 Bytes		→ 1 Byte	1 Byte	14 Bytes	2 Bytes

-Card no.

-Password: Appends 0xFF if insufficient or no password is provided.

-The password is compressed by combiing 2 digits into 1 byte.

- ":", 【Colon】is used to separate the card number from the password

- Provide 128 time shceuldes (0~127).

820PxF only one valid card is allowed.

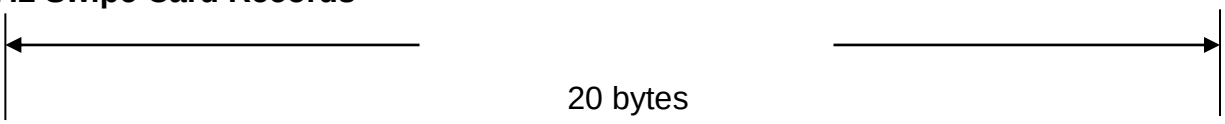
-Status description:

Bit0	0=Valid card 1=Blacklist
Bit1	0=Don't need password 1=Password is required
Bit2	0=Do need to check holiday 1=Check holiday
Bit3	0=Don't need to check time schedule 1=Check time schedule
Bit4	Reserved
Bit5	Reserved
Bit6	820PxF 0: Double matching on B group 1: Double matching on A group
Bit7	0: Not use voice 1: Use voice

Name: RAC-820PxF doesn't support display function. If there is no name information, please input 0x20, total 14 bytes.

System reservation: 2 bytes.

7.2 Swipe Card Records



Length	Status1	Status2	Date/Time	Duty Shift	Card No.
1 byte		1byte	4 bytes	1 byte	1-13 bytes

- Length: 4 bit, indicates the length of card no.
- Status1:

Bit0	0=requires input of card number 1=swipe card
Bit1	0=inside reader 1=outside reader
Bit2	0=General card no. 1=DesFire serial no. (7 bytes/ hexadecimal code)
Bit3	0=Swipe card record 1=Event record

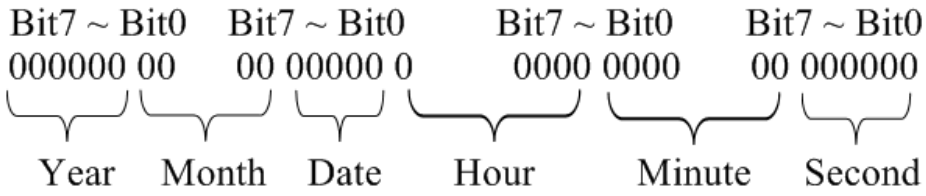
- Status 2:

Bit0~7	0 = Swipe of valid card 1 = Master card 2 = Disarm code 3 = Duress code 5 = Temporary card 6 = Black list 8 = Failure of fingerprint matching 9 = Sucess of fingerprint matching 10 = Guest card 11 = Guest card(unlimited usage) 20 = Can not foundd 21 = Authorization failed (Wrong time schedule) 30 = Unlock door by password 31 = Operation error 61 = Swipe of valid card + correct input of password 62 = Swipe of valid card + incorrect input of password 63 = Input of card no. + password

	67 = Anti-passback error 75 = Error length of compare valid digits 76 = Double matching error

- Date and time: Total of 4 byte(transmission starts with Lo byte),
  0-5 bit contains seconds
  6-11 bit contains minute
  12-16 contains hours
  17-21 contains date,
  22-25 contains month
  26-31 contains year (2000+N)

- Time (4byte) Byte1,Byte2,Byte3,Byte4



- Duty Shift:

0=BLANK	1=DUTY ON
2=DUTY_OFF	3=BREAK OUT
4=BREAK_IN	5=OT_START
6=OT_END	

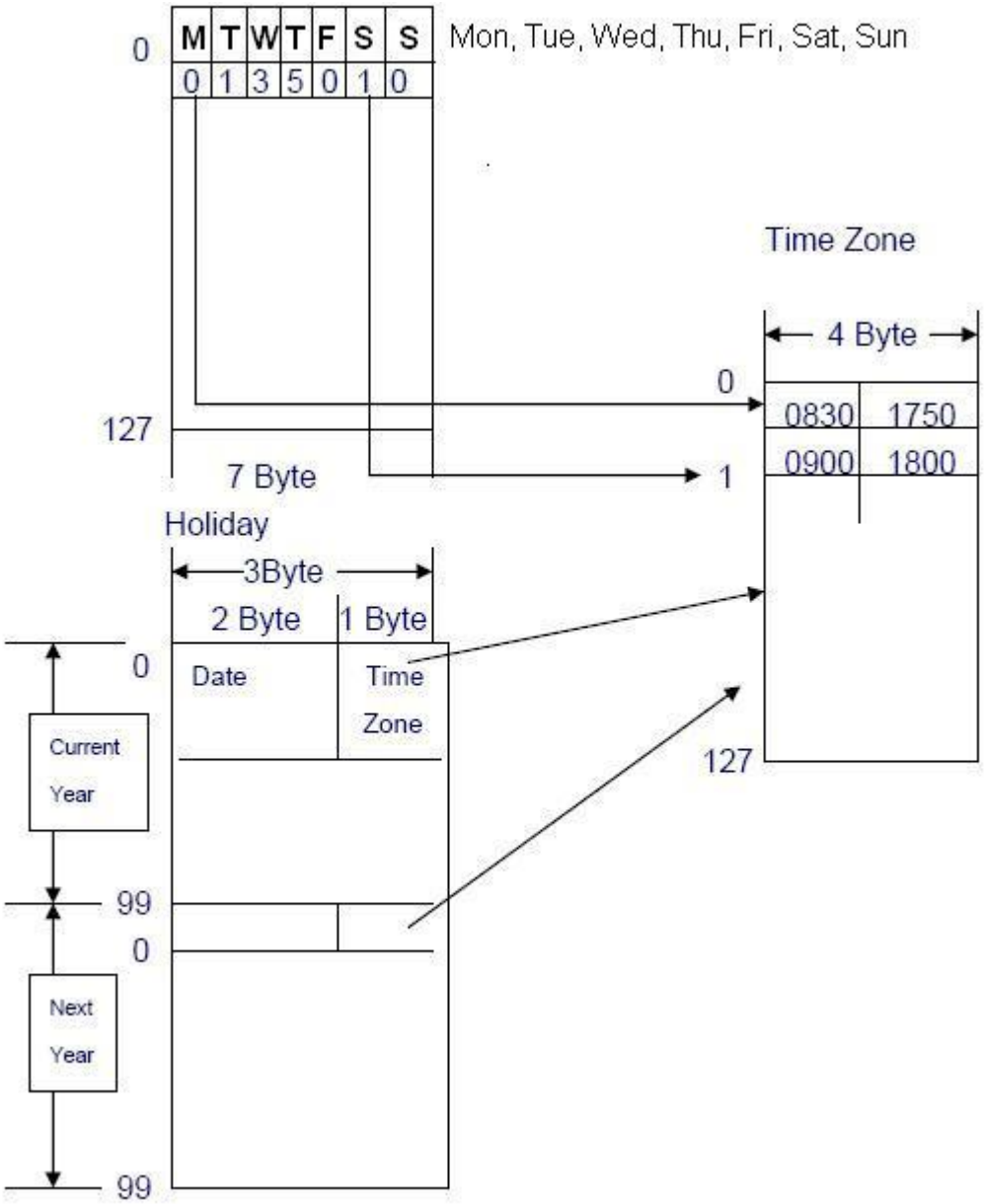
- Card number stored is base on length.

＊ When polling with duty shift information, kindly use 1FH command. 10H Command does not support polling with duty shift information.

7.3 Time Schedule

Time Schedule, Time Zone, Holiday

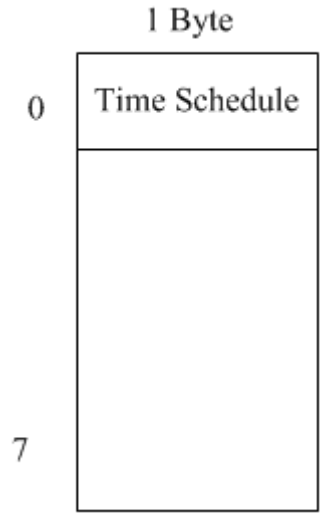
Time Schedule：



＊ Note: FF is appended if time zone, time schedule and holiday schedule is not provided

7.4 Unrestricted Time Schedule

8 groups



7.5 Event Code's Return Value (Sensor / Relay)

0120H	Duress code recognized
0122H	Disarm code deactivates alarm
0123H	Master card(code) modified
0301H	Tamper → Case Sensor
0306H	Cold startup
0307H	Reset
0311H	Door forced open → Door Sensor (Door has been breakthrough)
0312H	Door prop alarm
0313H	Deactivate door prop alarm
0318H	Door opened → Valid access
0319H	Door opened → Door relay closed
031AH	Exit button pressed → Push button
0401H	Memory full
041BH	System initialization

7.6 Flash Memory Allocation

(Address 0000H~00FFH：Total 256Bytes).

Address 0000H~0013H are system parameter.

Length	ADD	Description	Default	Remarks
1	0x0000	Communication: Bit7 = 1:TCP/IP Bit0~6: reserved	0x80	
1	0x0001	Reader format Reader format is determined by hardware and corresponding firmware. Bit1~3: reserved Bit4~7: slave reader 0=T1/T2 1=Wiegand26 (8 bytes) 2=Wiegand26 (10 bytes) 3=Wiegand34 (10 bytes) 4~15: reserved	0x01	
4	0x0002	IP	172.16.250.100	
4	0x0006	SubMask	255.255.0.0	
4	0x000A	Gateway	0.0.0.0	
2	0x000E	Port	4660	Lo Byte,Hi Byte
2	0x0010	Max. valid card number	0x80, 0x46	Lo Byte,Hi Byte 0x4680=18048 set
1	0x0012	Controller ID	0x01	
1	0x0013	Controller mode 1= Valid card	0x01	
1	0x0014	Control status setup Bit0: Slave reader doesn't need to check time schedule 0= Do not need to check time schedule 1= Check time schedule	0x00	

		Bit1:Indoor/Outdoor mode selection 0=Outdoor 1=Indoor Bit2:Enable Alarm Relay by blacklist 0=Disable (Disable Relay) 1=Enable (Enable Relay) Bit3:When slave reader is in card swipe + PIN mode, no password is required 0=Disable (Need password) 1=Enable (Don't need password) Bit4:Display card number Not display card number Bit5:Reserve invalid card records or not 0=Yes 1=No Bit6:Overwrite stored data or not 0=Overwrite 1= Noto overwrite, alarm warning when records reach to 90%. Bit7: Elable alarm relay when memory is full. 0=Disable (Disable Relay) 1=Enable (Enable Relay)		
1	0x0015	Year of the holiday schedule	0x16	
10	0x0016	Mifare setup (Only use Write Table 54 to set KeyA/KeyB value)	0x00,0x10,0x00,0x00, 0xFF,0xFF,0xFF, 0xFF,0xFF,0xFF	Refer to Write Table 54
2	0x0024	Retrieve valid digits length of valid card	0x01, 0x00	
2	0x0026	Compare valid digits length of valid card	0x00, 0x00	
1	0x0028	0=Anti Disable 1=Anti Enable	0x00	
1	0x0029	Language selection 0 = English	0x00	

		1 = Traditional Chinese 2 =Simplified Chinese		
1	0x002A	Sensor Status (1=Close,0=Open) Bit0 = 0 (Case Sensor) Bit1 = 0 (Door Sensor) Bit2 = 1 (Push Button) Bit3~7 = Reserved	0x04	
1	0x002B	Case Sensor enable Alarm Relay or not Bit0 = Buzzer Bit1 = Alarm Relay action Bit2~7 = Rserved	0x00	
1	0x002C	Door Sensor detection time 0= Disable 1~255=Detectiion time1~255 sec.	0x00	
1	0x002D	Door Sensor enable alarm or not Bit0 = Buzzer Bit1 = Alarm Relay action Bit2~7 = Reserved	0x01	
1	0x002E	Max. error times of swipe card 0= Close 1~255=Error times	0x00	
1	0x002F	Max. times of errors enable terminating (sec.) 0= Close 1~255=1~255sec.	0x00	
14	0x0030	Master Card (0xFF is appended to unused space)	‘30191000’	
1	0x003E	Fingerprint device status setting Bit0 = Fingerprint device is on 1:1 mode. (0=Auto,1=1:1) Bit1 = Fingerprint Replace Master Card (0=Disable,1=Enable)	0x00	

<table>
<tr><td></td><td></td><td>Bit2 = Double matching
(0=Disable,1=Enable)
Bit3 = Voice status
(0=Enable, 1=Disable)
Bit4 = Second RS-485 mode
0=Polling slave fingerprint device
1=DVR
Bit5 = Card priority, no fingerprint
(0=Disable,1=Enable)
Bit6~7 = Reserved</td><td></td><td></td></tr>
<tr><td>1</td><td>0x004E</td><td>Controller parameter setup
Bit0: LCD duty shift automatically recover
0=Not recover
1=Recover
Bit1: Max. times of errors enable Alarm Relay
0=Disable (Disable Relay)
1=Enable (Enable Relay)
Bit2: Roller shutter mode
0:Access control mode
1: Roller shutter mode
Bit3~7: Reserved</td><td>0x01</td><td>Relay 0 contains Relay of controller.
Relay1~2 contains Relay 1~2 of ACU-30</td></tr>
<tr><td>1</td><td>0x004F</td><td>Relay mode selection
<table><tr><td></td><td>Relay0</td><td>Relay1</td><td>Relay2</td></tr><tr><td>Mode 0</td><td>Door</td><td>Door</td><td>Alarm</td></tr><tr><td>Mode1</td><td>Bell</td><td>Door</td><td>Alarm</td></tr></table></td><td>0x00</td><td>Relay 0 contains Relay of controller.
Relay1~2 contains Relay 1~2 of ACU-30</td></tr>
<tr><td>14</td><td>0x0050</td><td>Duress Code
(0xFF is appended to unused space)</td><td>‘1190’</td><td></td></tr>
<tr><td>1</td><td>0x005E</td><td>Repeat activate mode of Alarm Relay
Bit0~5
0 = Activate Alarm Relay once
1~63= Inactivate Alarm Relay after activate alarm for few minutes</td><td>0x00</td><td>It will only work in Latch mode of Alarm Relay</td></tr>
</table>

		Bit6~7 = Reserved		
1	0x005F	In repeat activate mode,Relay activate time and repeat activate frequency Bit0~4 After 1~31 minutes, activate alarm relay again Bit5~7 1~7 = frequency	0x00	It will only work in Latch mode of Alarm Relay
2	0x0060	Relay0 setup (Door) Bit0~11 = Time of Pulse 0= Refer to other mode 1~4095=0.1~409.5 sec. Bit12~14 = Reserved Bit15 = Other mode (Time o fPulse is 0 sec.) 0=Toggle 1=Latch (unsupported)	0x28,0x00	Lo Byte,Hi Byte 0x0028=40 40*0.1 sec.=4 sec.
2	0x0062	Relay1 setup (Alarm) Bit0~11 = Time of Pulse 0=Refer to other mode 1~4095=0.1~409.5 sec. Bit12~14 = Reserved Bit15 = Other mode (Time o fPulse is 0 sec.) 0=Toggle 1=Latch	0x00,0x80	Lo Byte,Hi Byte
1	0x004E	Controller status setup Bit0: LCD duty shift automatically recover 0=Not recover 1=Recover Bit1: Max. times of errors enable Alarm Relay 0=Disable (Disable Relay) 1=Enable (Enable Relay) Bit2: Roller shutter mode 0: Access control mode 1: Roller shutter mode	0x01	

<table>
<tr><td></td><td></td><td>Bit3~7: Reserved</td><td></td><td></td></tr>
<tr><td>1</td><td>0x004F</td><td>Relay mode selection
<table><tr><td></td><td>Relay0</td><td>Relay1</td><td>Relay2</td></tr><tr><td>Mode 0</td><td>Door</td><td>Door</td><td>Alarm</td></tr><tr><td>Mode 1</td><td>Bell</td><td>Door</td><td>Alarm</td></tr></table></td><td>0x00</td><td>Relay 0 contains Relay of controller. Relay1~2 contains Relay 1~2 of ACU-30</td></tr>
<tr><td>1</td><td>0x005E</td><td>Repeat activate mode of Alarm Relay
Bit0~5
0 = Activate alarm relay once
1~63= Inactivate alarm after activate alarm for few minutes
Bit6~7 = Reserved</td><td>0x00</td><td>It will only work in Latch mode of alarm relay.</td></tr>
<tr><td>1</td><td>0x005F</td><td>In repeat activate mode,Relay activate time and repeat activate frequency
Bit0~4
After 1~31 minutes, activate alarm relay again
Bit5~7
1~7 = frequency 1~7</td><td>0x00</td><td>It will only work in Latch mode of alarm relay.</td></tr>
<tr><td>2</td><td>0x0060</td><td>Relay0 setup (Door)
Bit0~11 = Time of Pulse Pulse
0=Refer to other mode
1~4095=0.1~409.5 sec.
Bit12~14 = Reserved
Bit15 = Other mode (Time of Pulse is 0 sec.)
0=Toggle
1=Latch (unsupported)</td><td>0x28,0x00</td><td>Lo Byte, Hi Byte
0x0028=40
40*0.1 sec.=4sec.</td></tr>
<tr><td>2</td><td>0x0062</td><td>Relay1 setup (Alarm)
Bit0~11 = Time of Pulse
0=Refer to other mode
1~4095=0.1~409.5 sec.
Bit12~14 = Reserved
Bit15 = Other mode (Time of Pulse</td><td>0x00,0x80</td><td>Lo Byte,Hi Byte</td></tr>
</table>

		is 0 sec.) 0=Toggle 1=Latch		
1	0x005E	Repeat activate mode of alarm relay Bit0~5 0 = Activate alarm relay once 1~63= Inactivate alarm after activate alarm for few minutes. Bit6~7 = Reserved	0x00	Alarm Relay Latch
1	0x0069	1=Enable/ 0=Disable Bit0 Mifare Card Enable/Disable Bit1 CPU Card Enable/Disable Bit2-7 Reserved	0x03	Mifare/CPU Card Enable (PMF Only)
1	0x006A	RTC fine tune Bit0-6: Adjust 1 sec by several hours Bit7:0=Backward,1=Forward	0x08	Backward 1 sec every 8 hours
1	0x006B	Re-swipe card check time 0=Disable Rnage 1~255 sec.	0x00	

7.7.1 Time Schedule

(Address: 0100H~047FH: total 896 bytes)

128*7	0100H	Time Sechdule 128 groups * 7 bytes		

Byte1：Corresponding time schedule of Monday (00H~7FH 128 groups)

Byte2：Corresponding time schedule of Tuesday (00H~7FH 128 groups)

……

Byte7：Corresponding time schedule of Sunday (00H~7FH 128 groups)

＊ FFH is appended to unused space

7.7.2 Time Zone

(Address 0480H~067FH：Total 512 bytes)

128*4	0480H	Time Zone 128 groups*4 byte		

Start Time: 2 byte, First byte is hour. Second byte is minute.

Example: 13：45 => 13H,45H (BCD format)

End Time: 2 byte, First byte is hour. Second byte is minute.

Example: 20：12 => 20H,12H (BCD format)

＊ FFH is appended to unused space.

7.7.3 Holiday Schedule

(Address 0680H~08D7H：Total 600 bytes)

200*3	0680H	Holiday schedule 200 groups * 3 bytes		Current Year has 0~99 groups. Next year has 100~199 groups

Holiday Date：2 bytes, First byte is month. Second byte is date.

Example : December 25 => 12H,25H (BCD format)

Corresponding time schedule：1 byte

Example: Second Time schedule => 01H (00H~7FH 128 groups)

＊ FFH is appended to unused space.

7.7.4 Unrestricted Time Schedule

(Address 08D8H~08DFH：Total 8 bytes)

8*1	08D8H	8 groups time schedule of continuted open door		

＊ FFH is appended to unused space.

7.7.5 Alarm Time Schedule

(Address 08E0H~097FH：Total 160 bytes)

32*5	08E0H	Alarm time schedule 32 groups * 5 bytes		

Start Time：2 byte, First byte is hour. Second byte is minute.

Example : 08：30 => 08H,30H (BCD format)

Action Time：2 byte, First byte is minute. Second byte is second.

Example : 00：12 => 00H,12H (BCD format)

Weekday：1 byte

bit7： Reserved, bit6~0： Sun/Sat/Fri/Thu/Wed/Tue/Mon

＊ FFH is appended to unused space.

7.7.6 Anti Reset Time Schedule

(Address 0980H~099FH：Total 32 bytes)

16*2	0980H	Time clear Anti sign,16 groups* 2 bytes		

Time：2 bytes, First byteis hour. Second byte is minute.

Example 23：50 => 23H,50H (BCD format)

＊ FFH is appended to unused space.

7.7.7 Duty Shift Time Schedule

(Address 09A0H~09FFH：Total 96 bytes)

32*3	09A0H	32 groups* 3Byte		

Group	Time (2 byte)	Duty Shift (1 byte)
First group, address 0090H	HH：MM	Class
2th group, address 0093H	HH：MM	Class
3rd group, address 0096H	HH：MM	Class
.	.	.
.	.	.
.	.	.
32 group, address 00EDH	HH：MM	Class

Time：2 bytes, First byte is hour. Second byte is minute.

Example : 08：10 => 08H,10H (BCD format)

Duty Shift:1 byte

0=BLANK	1=DUTY ON

2=DUTY_OFF	3=BREAK OUT
4=BREAK_IN	5=OT_START
6=OT_END	

FFH is appended to unused space.

7.7.8 Request Password Time Schedule

(Address: 0x0A80~0x0A87: Total 8 bytes)

8*1	0x0A80	Request Password Time Schedule 8 groups		

＊ 0xFF is appended to unused space.

7.7.9 Alarm Time Schedule

(Address 0x0A88~0x0A8B: total 4 bytes)

4*1	0x0A88	Alarm time schedule 1 group		

2 bytes	2 bytes
Start time HHMM	End time HHMM

Start Time: Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 13:25 => 13H,25H (BCD format)

End Time: Takes up 2 bytes, first byte contains hour, second byte contains minute.

Example: 17:01 => 17H,01H (BCD format)

＊ 0xFF is appended to unused space.

7.7.11 Swiped Records

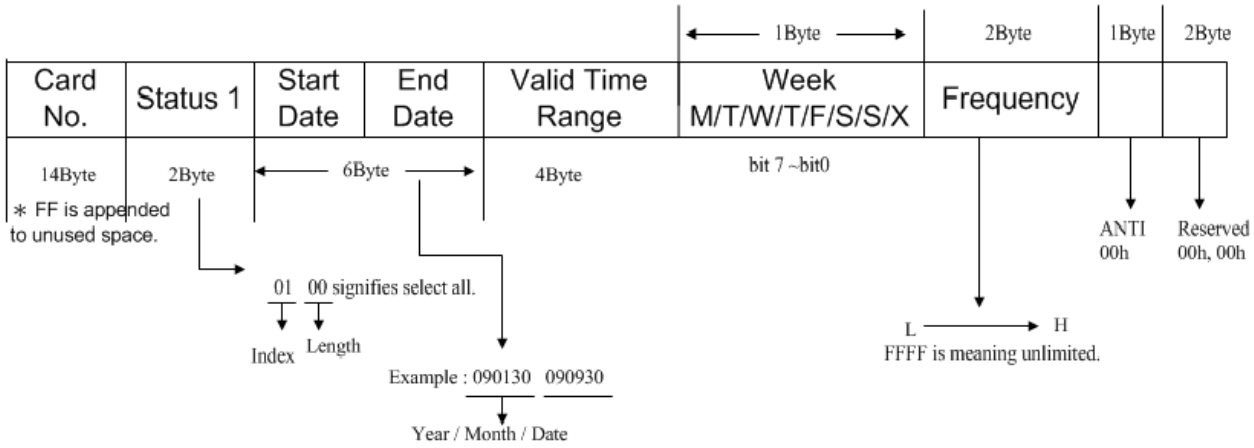
(Address 028000H~07FFFFH)

7.7.12 SRAM Memory Allcation

(R/W Table 15,16)

Guest Card: 50 groups (Address 0000H ~ 063FH : Total 1600 bytes)

50*32	0000H	50 groups Guest Card * 32 bytes		



7.7.13 Event Records 200 groups

(Address0640H~0C7FH：Total 1600 bytes)

200*8	0640H	Event records 200 groups* 8 bytes		