iData UHF module interface document

Document version	SDK version	Modify records	Participant	Time
1.0	1.4.01	UHF interface list	software team	2022.7.22
1.1	1.4.01	Added temperature acquisition interface	software team	2022.8.29
1.2	1.4.02	Add RM module	software team	2022.10.20
1.3	1.4.02	Add fan interface	software team	2023.6.5
1.4	1.4.03	Increase read and write power to set interfaces separately	software team	2023.8.23

1. boolean powerOn();

Function	Power on the module and turn on the UHF device
Function prototype	boolean powerOn()
Describe	Open the serial port and power on the module. The baud rate of the serial port is 115200.
Return	Success: true Fail: false

2. boolean powerOff();

Function	Power off the module and turn off the UHF device
Function prototype	boolean powerOff()
Describe	Close the serial port and power off the module
Return	Success: true Fail: false

3. int powerGet();

Function	Get power
Function prototype	int powerGet()
Describe	Get the current module power
Return	Success: Return current power Fail: -1

4. boolean powerSet(int power) ;

Function	Set power
Function prototype	boolean powerSet(int power)
Describe	Set power Parameter: The power that needs to be set

Return	Success：true Fail：false

5. boolean frequencyModeSet(int freMode);

Function	Set zone frequency
Function prototype	boolean frequencyModeSet(int freMode)
Describe	Set zone frequency parameter: freMode: 0:China(840-845MHz) 1:China(920-925MHz) 2: Europe (865-868MHz) 3: United States (902-928MHz) 5: UM2 customized frequency point (note only supported by UM2 module equipment)
Return	Success：true Fail：false

6. int frequencyModeGetNotFixedFreq();

Function	Get area frequency
Function prototype	int frequencyModeGetNotFixedFreq()
Describe	Get area frequency
Return	Success: 0:China(840-845MHz) 1:China(920-925MHz) 2: Europe (865-868MHz) 3: United States (902-928MHz) 5: UM2 customized frequency point (note only supported by UM2 module equipment) Fail: -1

7. boolean startInventoryTag();

Function	Start taking inventory of labels
Function prototype	boolean startInventoryTag()
Describe	Start taking inventory of labels
Return	Success：true Fail：false

8. boolean stopInventory();

Function	Stop counting labels
Function prototype	boolean stopInventory()
Describe	Stop counting labels
Return	Success：true Fail：false

9. String[] readTagFromBuffer();

Function	Get the tag data during the inventory
Function prototype	String[] readTagFromBuffer()
Describe	Get buffer data
Return	Success: UM, SLR module String[0]: stores TID data. If the inventory is read-only EPC, it is null. String[1]:EPC data String[2]:RSSI data RM module String[0]: reserved, null String[1]:EPC data

	String[2]:RSSI data String[3]:Temperature failed: null

10.boolean filterSet(int bank, int ptr, int cnt, String data, int save); (modification)

Function	Set label filter conditions
Function prototype	boolean filterSet(int bank, int ptr, int cnt, String data, int save)
Describe	Set label filtering conditions Parameters: bank: buckets to filter 0:RESERVED 1:EPC 2:TID 3:USR ptr: filter start address (bit type) cnt: Filter data length (bit type) data: data to filter Note: the RM module can only filter the EPC area temporarily and the first three parameters are invalid save: whether to save after power off 1-save, 0-not save (SLR, RM modules do not support)
Return	Success: true Fail: falseS

11.boolean lockMen(String accessPwd, int bank, int ptr, int cnt, String data, int bank1, int locktype); (modification)

Function	Lock tag
Function prototype	boolean lockMen(String accessPwd, int bank, int ptr, int cnt, String data, int bank1, int locktype)
Describe	Lock tag Parameters: accessPwd: Access Password of label RESERVED area

	bank: filter storage area for tags 1:EPC 2:TID 3: USER ptr: starting address of filtering (bit) cnt: filter length (bit) data: data that needs to be filtered bank1: area that needs to be locked 0: Kill(RFU) 1:Access(RFU) 2:EPC 3:TID 4:USER locktype: whether the lock type is a normal lock or a permanent lock (only supported by the UM module) 0: Normal lock; 1: Permanently locked; (After the tag is permanently locked successfully, it cannot be unlocked) Note: The RM module can only filter the EPC area bank, ptr, cnt parameters are invalid
Return	Success：true Fail：false

12.boolean unlockMen(String accessPwd, int bank, int ptr, int cnt, String data, int bank1, int locktype); (modification)

Function	Unlock tag
Function prototype	boolean unlockMen(String accessPwd, int bank, int ptr, int cnt, String data, int bank1, int locktype)
Describe	Unlock tag Parameters: accessPwd: Access Password of label RESERVED area bank: filter storage area for tags

	1:EPC 2:TID 3: USER ptr: starting address of filtering (bit) cnt: filter length (bit) data: data that needs to be filtered bank1: area that needs to be locked 0: Kill(RFU) 1:Access(RFU) 2: EPC 3:TID 4:USER locktype: whether the lock type is normal unlocking or permanent unlocking (only supported by UM module) 0: Normal unlock; 1: Permanently unlocked; (After the tag is permanently unlocked successfully, it cannot be locked) Note: The RM module can only filter the EPC area bank, ptr, cnt parameters are invalid
Return	Success：true Fail：false

13.boolean writeDataToEpc(String accessPwd, int ptr, int cnt, String data)

Function	Write data to the tag's EPC area (without filtering)
Function prototype	boolean writeDataToEpc(String accessPwd, int ptr, int cnt, String data)
Describe	Write data to the EPC area of the tag (without filtering) parameter: accessPwd: Access Password of label RESERVED area ptr: starting position to start writing (Word type)

	cnt: length of written data (Word type, greater than 0) data: Data to be written (hexadecimal)
Return	Success：true Fail：false

14.boolean writeTag(String accessPwd, int bank, int ptr, int len, String data, int bank1, int ptr1, int len1, String data1);

Function	Write tag data
Function prototype	boolean writeTag(String accessPwd, int bank, int ptr, int len, String data, int bank1, int ptr1, int len1, String data1)
Describe	Write tag data Parameters: accessPwd: Access Password of label RESERVED area bank: filtered storage area 1:EPC 2:TID 3: USER ptr: starting address of filtering (bit) len: filter length (bit) data: data that needs to be filtered bank1: target area for write operations 0: RFU 1:EPC 2:TID 3: USER ptr1: starting address (Word) len1: length (Word) data1: data to be written Note: The RM module can only filter the EPC area bank, ptr, cnt parameters are invalid
Return	Success：true Fail：false

15.String readTag(String accessPwd, int bank, int ptr, int len, String data, int bank1, int ptr1, int len1);

Function	Read tag data
Function prototype	String readTag(String accessPwd, int bank , int ptr, int len, String data, int bank1, int ptr1, int len1)
Describe	Read tag data Parameters: accessPwd: Access Password of label RESERVED area bank: filtered storage area 1:EPC 2:TID 3: USER ptr: starting address of filtering (bit) len: filter length (bit) data: data that needs to be filtered bank1: target area for read operations 0: RFU 1:EPC 2:TID 3: USER ptr1: starting address (Word) len1: length (Word) Note: The RM module can only filter the EPC area bank, ptr, cnt parameters are invalid
Return	Success: The label data was read Fail：null

16.boolean killTag(String killPwd, int bank, int ptr, int cnt, String data);

Function	Kill tag
Function prototype	boolean killTag(String killPwd, int bank, int ptr, int cnt, String data)
Describe	Kill tag Parameters:

	killPwd: Kill password for the label RESERVED area bank: Filtered storage area 1：EPC 2：TID 3：USER ptr: the starting address of filtering (bit) cnt: length of filtering (bit) data: the data to be filtered Note: RM module can only filter EPC area bank, ptr, cnt parameters are invalid
Return	Success：true Fail：false

17.String hardwareVerGet();

Function	Get hardware version number
Function prototype	String hardwareVerGet()
Describe	Get hardware version number
Return	Success: Hardware version number Fail:""

18.String firmwareVerGet();

Function	Get firmware version number
Function prototype	String firmwareVerGet()
Describe	Get firmware version number
Return	Success: firmware version number Fail: ""

19.boolean updateFirmware(String filePath, String fileName);

Function	Upgrade firmware
Function prototype	boolean updateFirmware(String filePath, String fileName)
Describe	Upgrade firmware (SLR module is only supported by SLR7100 module) Parameters: filePath: the absolute path where the firmware file is stored in the SD card fileName: firmware file name
Return	Success: true Fail: false

20.boolean readTagModeSet(int mode, int startaddr, int wordcnt, int ifSave); (modification)

Function	Set the type of data returned by label inventory
Function prototype	boolean readTagModeSet(int mode, int startaddr, int wordcnt, int ifSave)
Describe	Set the type of data returned by label inventory Parameters: mode: 0: Only read tag EPC 1: Read tag EPC and TID 2: Read tags EPC and USER 3: Indicates reading EPC, TID and USR data at the same time (only supported by UM module) 4: Indicates reading EPC, TID and RFU data at the same time (only supported by UM module) 5: Indicates reading EPC and RFU data at the same time (only supported by UM module) startaddr: starting address (only supported by UM module). This parameter is used for inventory USR area. Just set 0 when reading TID.

	wordcnt : word length (only supported by UM module). This parameter is used for inventory USR area. Just set 0 when reading TID. ifSave : whether to save when power off (only supported by UM module) 1: Save after power off 0: No saving when power off Note: RM module does not support UM2 module currently.
Return	Success：true Fail：false

21.int readTagModeGet();

Function	Get the type of data returned by the current label inventory
Function prototype	int readTagModeGet()
Describe	Get the type of data returned by the current label inventory
Return	Success: 0: Only read tag EPC 1: Read tag EPC and TID 2: Read tags EPC and USER 3: Indicates reading EPC, TID and USR data at the same time (only supported by UM module) 4: Indicates reading EPC, TID and RFU data at the same time (only supported by UM module) 5: Indicates reading EPC and RFU data at the same time (only supported by UM module) Fail: other Note: RM module does not support, UM2 module does not support currently.

22.boolean sessionModeSet(int sessionValue);

Function	Set session mode
Function prototype	boolean sessionModeSet(int sessionValue)
Describe	Set session mode Parameters: sessionValue: 0:S0 1:S1 2:S2 3:S3 Note: RM module is not supported yet
Return	Success: true Fail: false

23.int sessionModeGet();

Function	Get session mode
Function prototype	int sessionModeGet()
Describe	Get session mode
Return	Success: 0:S0 1: S1 2: S2 3:S3 Fail: -1 Note: RM module is not supported yet

24.boolean inventoryModelSet(int mode, boolean save);

Function	Set the mode of label inventory
Function prototype	boolean inventoryModelSet(int mode, boolean save)

Describe	Set the mode of label inventory (only supported by UM module) Parameter: mode: 0: Multi-label mode 1: Quick mode 2: Low power consumption mode 3: Test mode 4: Power saving mode save: True: Save after power off False: not saved when power off
Return	Success: true Failure: false

Schema name	Function description	Advantages and Disadvantages	Recommended application scenarios
Multi-tab mode	Works at full speed, default S1, configurable Session, duty cycle 100% (the same tag will only be read 1-2 times per second)	Advantages: High disk completion rate Disadvantages: High power consumption, high heat generation	Suitable for massive label inventory, UHF performance comparison test, such as clothing store inventory, warehouse inventory
fast read mode	Works at full speed, default S0, configurable Session, duty cycle 100% (the same tag can be read dozens of times in one second)	Advantages: Fast tag reading rate Disadvantages: High power consumption	Single tag long-distance reading, such as power inspection, assembly line applications, fast-moving items

Ultra-low power mode	Adaptive work, default S2, session cannot be configured, automatically adjusts the duty cycle, a small number of tag points have a duty cycle of 10%, and a large number of tags have a duty cycle of 50%	Advantages: ultra-low power consumption, extremely low heat generation Disadvantages : low reading speed	A large number of multi-tag reading, such as cabinets, warehousing, logistics, and clothing stores
test mode	An upgraded version of Mode 1 multi-tab mode Works at full speed, defaults to S1 and S2, session cannot be configured, duty cycle 100% (the same tag will only be read 1-2 times per second)	Advantages: high disk completion rate, best multi-tag performance Disadvantages : high power consumption, high heat generation	Suitable for massive label inventory, UHF performance comparison test, such as clothing store inventory, warehouse inventory
Adaptive power saving mode	Adaptive work, default S0 and S1, session cannot be configured, automatically adjust the duty cycle, a small number of tags have a duty cycle of 30%, and a large number of tags have a duty cycle of 90%	Advantages: 1. The label inventory performance is basically the same as that of Mode 1; 2. The duty cycle is automatically adjusted to save power	It is recommended to select this mode by default and is suitable for most scenarios.

		and reduce heat generation.	

25.int inventoryModelGet();

Function	Get the pattern of label inventory
Function prototype	int inventoryModelGet()
Describe	Get the mode of label inventory (only supported by UM module)
Return	success: 0: Multi-label mode 1: Quick mode 2: Low power consumption mode 3: Test mode 4: Power saving mode Fail: -1

26.boolean inventoryWaitTime_Set(int scanTime, int waitTime, boolean save);

Function	Set the duty cycle of inventory time
Function prototype	boolean inventoryWaitTime_Set(int scanTime, int waitTime, boolean save)
Describe	Set the duty cycle of inventory time Parameters: scanTime: time of inventory waitTime: sleep time save: whether to save after powering off (not supported by the SLR module) True: Save after power off False: not saved when power off
Return	Success：true

	Fail：false

27.int[] inventoryWaitTime_Get();

Function	Get the duty cycle of inventory time
Function prototype	int[] inventoryWaitTime_Get()
Describe	Get the duty cycle of inventory time
Return	Success: int[0]: inventory time int[1]: sleep time Fail: null

28.int frequencyRange_Set(int save, int num, int[] freqlist, int frequency); (modification)

Function	Set frequency range
Function prototype	int frequencyRange_Set(int save, int num, int[] freqlist,int frequency)
Describe	Set frequency point range (up to 50) Parameters:: save: whether to save after power off 1: save 0: Do not save num: number of frequency points freqlist: jump screen table Frequency: Index of regional frequency (only meaningful for RM module)
Return	Success：1 Fail：other .

29.int[] frequencyRange_Get();

Function	Get frequency range
Function prototype	int[] frequencyRange_Get()
Describe	Get the frequency range (SLR module is only supported by SLR7100 module and UM module)
Return	Success: jump screen table Fail: null

30.String getUHFModuleType();

Function	Get module type
Function prototype	String getUHFModuleType()
Describe	Get module type (only supported by SLR module)
Return	Success: Return the model number of the SLR module

31.boolean slrInventoryModeSet(int mode);

Function	Set inventory mode (SLR module)
Function prototype	boolean slrInventoryModeSet(int mode)
Describe	Set inventory mode (SLR module) Parameters mode: 0: S0, asynchronous inventory mode (used by SLR5100 module) 1: S1, asynchronous inventory mode (no module is used yet) 2: Smart mode (recommended for SLR1200) (setting of duty cycle and Session is not supported) 3: Fast mode (used by E710 (SIM7100)) (Session and duty cycle cannot be set) 4: Normal mode (suitable for reading single tags, poor

	performance in multi-tag scenarios) 5: E7 smart mode (the smart temperature control mode of E710 cannot set the duty cycle and session)
Return	Success: true Failure: false

32. int slrInventoryModelGet() ;

Function	Get inventory mode (SLR module)
Function prototype	int slrInventoryModelGet()
Describe	Get inventory mode (SLR module)
Return	Success: 0: S0, asynchronous inventory mode (recommended for SLR5100 module) 1: S1, asynchronous inventory mode (no module is used yet) 2: Smart mode (recommended for other SLR modules) 3: Fast mode (recommended for E710 (SLR7100)) 4: Normal mode 5: E7 smart mode Fail: other

33.String getModuleTemp(); (new)

Function	Get module temperature
Function prototype	String getModuleTemp()
Describe	Get module temperature
Return	Success: module temperature; Fail: null;

34.boolean openFan(); (new)

Function	Turn on the fan
Function	boolean openFan()

prototype	
Describe	Turn on the fan Note: This only takes effect on machines with fans.
Return	Success: true; Fail: false;

35.boolean closeFan(); (new)

Function	Turn off the fan
Function prototype	boolean closeFan()
Describe	Turn off the fan Note: This only takes effect on machines with fans.
Return	Success: true; Fail: false;

36.boolean setReadWritePower(int readPower,int writePower); (new)

Function	Set read and write power
Function prototype	boolean setReadWritePower(int readPower,int writePower)
Describe	Set read power and write power separately Parameters: readPower: read power writerPower: write power
Return	Success: true; Fail: false;

37.int[] getReadWritePower(); (new)

Function	Get read and write power
Function prototype	int[] getReadWritePower()
Describe	Get read power and write power respectively
Return	Success: int[0]: read power; int[1]: write power; Fail: null;