RTS30xx USB PC CAMERA CONTROLLER
User API Document

Rev. 1.00
Jan 16, 2015

COPYRIGHT

DISCLAIMER

Realtek provides this document "as is", without warranty of any kind, neither expressed nor implied, including, but not limited to, the particular purpose. Realtek may make improvements and/or changes in this document or in the product described in this document at any time. This document could include technical inaccuracies or typographical errors.

TRADEMARKS

Realtek is a trademark, of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

REVISION HISTORY

Revision	Release Date	Summary
1.00	2015/1/16	Preliminary release.

Table of Contents

1. Functions

1.1. Initialize/UnInitialize

HRESULT WINAPI **RvcLib_Initialize(BOOL Log, BOOL LibType)**
// RvcLib Initialize
// Log -[in] Generate the debug log.
// LibType -[in] Library type(Default:1; Vendor:0).
// Return Values: "0" indicates success, otherwise indicates failure

HRESULT WINAPI **RvcLib_UnInitialize(BOOL LibType)**
// RvcLib UnInitialize
// LibType -[in] Library type(Default:1; Vendor:0).
// Return Values: "0" indicates success, otherwise indicates failure

1.2. UVC_OPEN

DWORD UVC_Open(DWORD & index);
// Open camera device by index.
// index - [in] Device index
// Return Values: "0" indicates failure, otherwise indicates success.

1.3. UVC_Close

BOOL UVC_Close(DWORD devInst);
// Release device handle.
// devInst - [in] Device handle, You can retrive device hanle by UVC_Open.
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.4. *UVC_ReadI2CReg8*

BOOL UVC_ReadI2CReg8(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
// Read I2C data.
// devInst - [in] Device handle
// dwAddr - [in] Address
// uLength - [in] Buffer length
// lpData - [out] retrieve buffer from I2C bus
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.5. UVC_ *WriteI2CReg8*

BOOL UVC_WriteI2CReg8(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
// Write I2C data.
// devInst - [in] Device handle
// dwAddr - [in] Address
// uLength - [in] Buffer length
// lpData - [out] retrieve buffer from I2C bus
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.6. UVC_ReadI2CGeneric

BOOL UVC_ReadI2CGeneric(DWORD devInst, BYTE byI2CSlaveAddr, UINT uLength, LPBYTE lpData);
// Read I2C generic data.
// devInst - [in] Device handle
// byI2CSlaveAddr - [in] Slave ID
// uLength - [in] Buffer length
// lpData - [out] retrieve buffer from I2C bus
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.7. UVC_WriteI2CGeneric

BOOL UVC_WriteI2CGeneric(DWORD devInst, BYTE byI2CSlaveAddr, UINT uLength, LPBYTE lpData);
// Write I2C generic data.
// devInst - [in] Device handle
// byI2CSlaveAddr - [in] Slave ID
// uLength - [in] Buffer length
// lpData - [out] retrieve buffer from I2C bus
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.8. UVC_GetVIDPIDRevSN

BOOL UVC_GetVIDPIDRev(unsigned short* pVID, unsigned short* pPID, unsigned short* pRev, string pSN);
// Get camera VID/PID/Rev information.
// pVID - [out] Retrieve Vendor ID string.
// pPID - [out] Retrieve Product ID string.
// pRev - [out] Retrieve Revision string.
// pSN - [out] Retrieve Serial Number string.
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.9. UVC_ReadMem

BOOL UVC_ReadMem(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);

// Read AISC data from backend IC.
// devInst - [out] Device handle.
// dwAddr - [out] Read data address.
// uLength - [out] Read data length.
// lpData - [out] Retrieve data from backend.
// Return Values: "TRUE" indicates success, otherwise indicates failure

1.10. UVC_WriteMem

BOOL UVC_WriteMem(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
// Write AISC data to backend IC.
// devInst - [out] Device handle.
// dwAddr - [out] Read data address.
// uLength - [out] Read data length.
// lpData - [out] Retrieve data from backend.
// Return Values: "TRUE" indicates success, otherwise indicates failure

2. Sample Code

```
// test_sample.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include "afxwin.h"
#include <iostream>

using namespace std;

typedef BOOL(*pfn_RvcLib_Initialize)(BOOL bInternalLog, BOOL bLibType);
typedef BOOL(*pfn_RvcLib_UnInitialize)(BOOL bLibType);
typedef DWORD(*pfn_UVC_Open)(DWORD index);
typedef BOOL(*pfn_UVC_Close)(DWORD devInst);
typedef BOOL(*pfn_UVC_WriteMem)(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
typedef BOOL(*pfn_UVC_ReadMem)(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
typedef BOOL(*pfn_UVC_ReadI2CGeneric)(DWORD devInst, BYTE byI2CSlaveAddr, UINT uLength, LPBYTE lpData);
typedef BOOL(*pfn_UVC_WriteI2CGeneric)(DWORD devInst, BYTE byI2CSlaveAddr, UINT uLength, LPBYTE lpData);
typedef BOOL(*pfn_UVC_GetVIDPIDRev)(DWORD devInst, WORD * wVID, WORD * wPID, WORD *wRevID);
typedef BOOL(*pfn_UVC_ReadI2CReg8)(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
typedef BOOL(*pfn_UVC_WriteI2CReg8)(DWORD devInst, DWORD dwAddr, UINT uLength, LPBYTE lpData);
typedef BOOL(*pfn_UVC_GetVIDPIDRevSN)(DWORD devInst, WORD * wVID, WORD * wPID, WORD *wRevID, string
*szSN);

int _tmain(int argc, _TCHAR* argv[])
{
	HINSTANCE hLib = NULL;
#ifdef DEBUG
	hLib = LoadLibrary(L"..\\Debug_dll32\\RvcLib.dll");
#else
	hLib = LoadLibrary(L"..\\Release_dll32\\RvcLib.dll");

#endif
	if (!hLib)
	{
		hLib = LoadLibrary(L"RvcLib.dll");
	}

	if (hLib)
	{
		pfn_RvcLib_Initialize   RvcLib_Initialize = (pfn_RvcLib_Initialize)GetProcAddress(hLib, ("RvcLib_Initialize"));
		pfn_RvcLib_UnInitialize RvcLib_UnInitialize = (pfn_RvcLib_UnInitialize)GetProcAddress(hLib,
("RvcLib_UnInitialize"));
		pfn_UVC_Open            pUVC_Open = (pfn_UVC_Open)GetProcAddress(hLib, ("UVC_Open"));
		pfn_UVC_Close           pUVC_Close = (pfn_UVC_Close)GetProcAddress(hLib, ("UVC_Close"));
		pfn_UVC_ReadMem             pUVC_ReadMem = (pfn_UVC_ReadMem)GetProcAddress(hLib,
("UVC_ReadMem"));
		pfn_UVC_WriteMem        pUVC_WriteMem = (pfn_UVC_WriteMem)GetProcAddress(hLib,
("UVC_WriteMem"));
		pfn_UVC_ReadI2CGeneric  pUVC_ReadI2CGeneric = (pfn_UVC_ReadI2CGeneric)GetProcAddress(hLib,
("UVC_I2CGenericRead"));
		pfn_UVC_WriteI2CGeneric pUVC_WriteI2CGeneric = (pfn_UVC_WriteI2CGeneric)GetProcAddress(hLib,
("UVC_I2CGenericWrite"));
		pfn_UVC_GetVIDPIDRev    pUVC_GetVIDPIDRev = (pfn_UVC_GetVIDPIDRev)GetProcAddress(hLib,
("UVC_GetDevVIDPID"));
```

```
		pfn_UVC_ReadI2CReg8		pUVC_ReadI2CReg8 = (pfn_UVC_ReadI2CReg8)GetProcAddress(hLib,
("UVC_ReadI2CReg8"));
		pfn_UVC_WriteI2CReg8	pUVC_WriteI2CReg8 = (pfn_UVC_WriteI2CReg8)GetProcAddress(hLib,
("UVC_WriteI2CReg8"));
		pfn_UVC_GetVIDPIDRevSN	pUVC_GetVIDPIDRevSN =
(pfn_UVC_GetVIDPIDRevSN)GetProcAddress(hLib, ("UVC_GetDevVIDPIDSN"));
		if (RvcLib_Initialize)
		{
			RvcLib_Initialize(0, 1);

			BOOL bResult = FALSE;
			DWORD devIndex = 0;
			DWORD devInst = 0;

			unsigned short pVID = 0;
			unsigned short pPID = 0;
			unsigned short pRev = 0;
			string strSN;
			BYTE Buf[2] = { 0x0 };

			devInst = pUVC_Open(devIndex);

			if (0 == devInst)
			{
				printf("Cannot find the Realtek camera device.\n");
				return 0;
			}

			if (pUVC_GetVIDPIDRevSN(devInst, &pVID, &pPID, &pRev, &strSN))
			{
				printf("Device Info : VID=%04X,PID=%04X,REV=%04X,SN=%s\n", pVID, pPID, pRev,
strSN.c_str());
			}

			//+ Turn on at even frame	Reg 0xD50F = 1
			Buf[0] = 1;//
			pUVC_WriteMem(devInst, 0xD50F, 1, Buf);
			//-

			// Close device handle.
			pUVC_Close(devInst);

			RvcLib_UnInitialize(1);
		}
	}
	return 0;
}
```