VTT-100
Software user Guide

IEI® Technology, Corp.
Version: V1.0.0.0
DOCID VTT-1000.doc
Date: 20110323

Menu

Update OS

Step1.

Prepare micro SD Card (FAT32 format),

Copy below files to SD Card。

(in CD\SD CARD update OS folder)

EBOOT.bin

logo.bmp

NK.bin

STEPLDR.nb0

update.enable

(ps: logo.bmp resolution 320x240, RGB true color 24bit,bitmap)

Step 2.

Insert SD Card and power on then will auto update OS and LOGO files, red progress bar mean updating, blue progress bar means normal booting

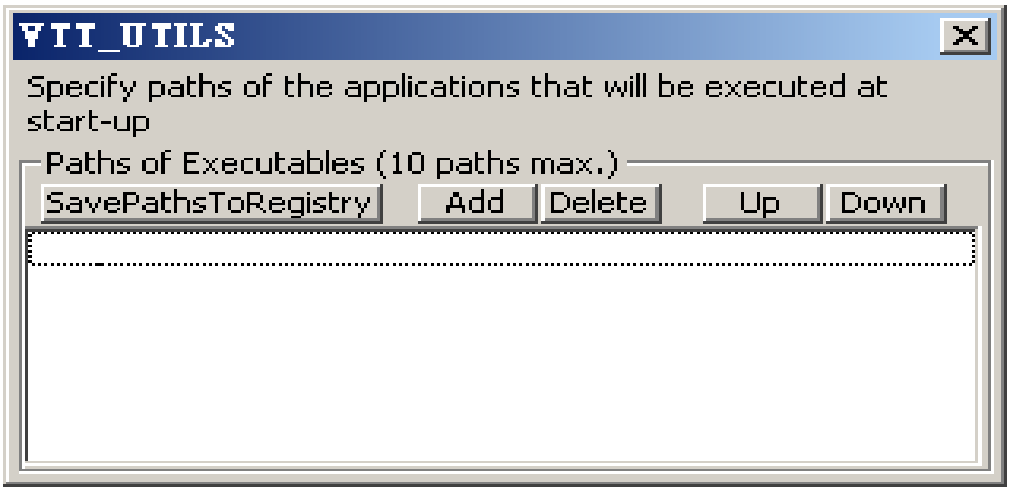
Step3.

first update will to do touch panel calibration, after calibration, take off SD card then reboot is finished.

VTT-1000 Tools

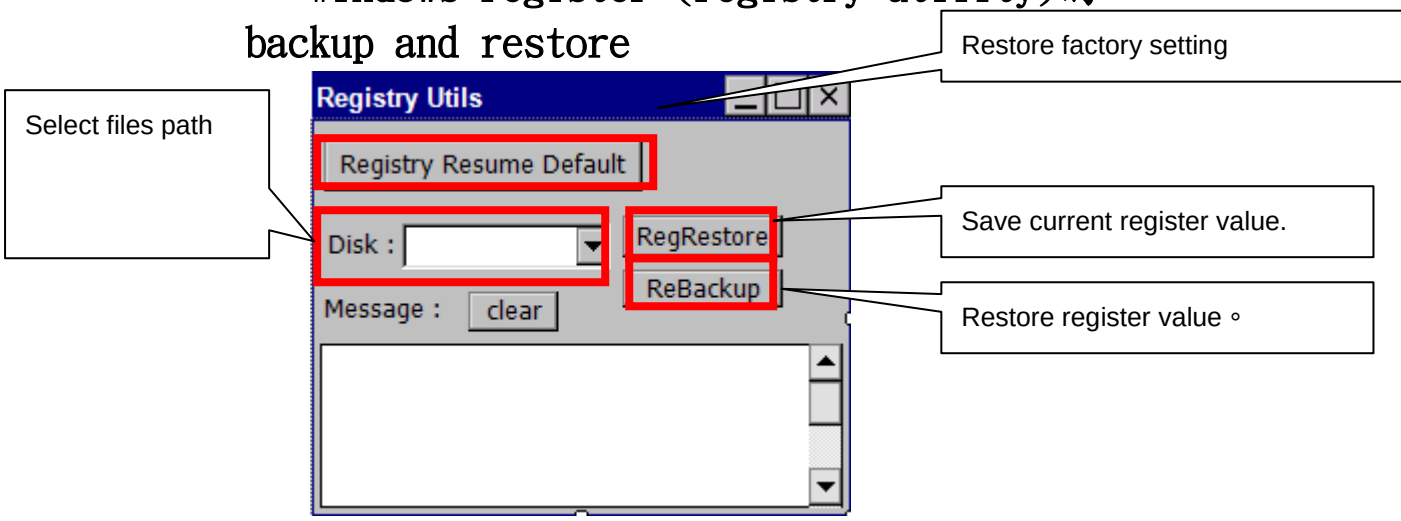
VTT_UTILS.exe

-Set Start up auto execute program



RegUtils.exe

Windows register (registry utility)的backup and restore



VTT-1000 SDK

Provide an API so that programmers can write applications consistent with the operating environment. Although APIs are designed for programmers, they are ultimately good for users because they guarantee that all programs using a common API will have similar interfaces. This makes it easier for users to learn new programs. For details, see the following with the Extension.

MINI USB Driver
Windows CE usb serial driver fo windows xp/2000

VTT1000Dll.dll
1. Digital IO API:
2. G-Sensor API
3. Program Key API
4. Temperature API
5. E-Compass API
6. COM API

CElcpFleetMangDll.dll
1. OBDII API
2. GPS API

Reference demp AP
Testing.exe
-DIO/COM PORT/ Program key/ Temperature/G Sensor/GPRS..etc

- ## Setting File (testing.ini)
- **Define 1 is true execute, 0 is false no execute:**

  autorun=1 (Auto execute test)

  log=1 (test and recording)

  burn=1 (Loop Test)

TestOBD_CE.exe
-Read OBDII PID data

VTT1000Dll.dll

1. Digital IO API:

- BOOL DIO_WRITE0(byte* pByteHighOrLow):

  Input : byte*, 1 byte data
  Output: void
  Return: BOOL, true or false

- BOOL DIO_WRITE1(byte* pByteHighOrLow):

  Input : byte*, 1 byte data
  Output: void
  Return: BOOL, true or false

- BOOL DIO_READ0(byte* pByteHighOrLow):

  Pin 1 read
  Input : void
  Output: byte*, 8 byte array buffer
  Return: BOOL, true or false

- BOOL DIO_WRITE1(byte* pByteHighOrLow):

  Pin 2
  Input : void
  Output: byte*, 8 byte array buffer
  Return: BOOL, true or false

2. COM(Serial) Port API:

- BOOL COM_Open(int nPort, int nBaud):

  Input : int, port number
  Input : int, boud rate
  Output: void
  Return: BOOL, true or false

- BOOL COM_Close():

  Input : void
  Input : void
  Output: void
  Return: BOOL, true or false

- BOOL COM_IsOpened():

  Input : void
  Output: void
  Return: BOOL, true or false

- int COM_ReadData(byte* pBufData, int iSize):

  input : int, read length
  output: byte*, byte array data
  return: int, read length

3. Temperature API:

- BOOL GetCurrTemperatureValue(DOUBLE* pDblTempValue):

Input : void
Output: DOUBLE*, temperature value
Return: BOOL, true or false

4. Program Key API:

- BOOL GetKeyPadValue(BYTE* pByteKeyNum):

Return : true if succeed, false if fail
Output : BYTE* pByteKeyNum
Input : void
*pByteKeyNum=1:press Key1, *pByteKeyNum=2:press Key2
*pByteKeyNum=3:press Key3, *pByteKeyNum=4:press Key4
*pByteKeyNum=5:press Key5, *pByteKeyNum=6:press Key6
*pByteKeyNum=7:press Key7, *pByteKeyNum=0:failed

5. E-Compass API:

- BOOL GetCompassModeValue(INT32* pIntModeValue):

Return : true if succeed, false if fail
Output : INT32* pIntModeValue
Input : void
*pIntModeValue=0:continuous mode, *pIntModeValue=1:single mode
*pIntModeValue=2:idle mode, *pIntModeValue=3:sleep mode
*pIntModeValue=-1:failed

- BOOL SetCompassModeValue(INT32* pIntModeValue):

Return : true if succeed, false if fail
Input : INT32* pIntModeValue
*pIntModeValue=0:continuous mode, *pIntModeValue=1:single mode
*pIntModeValue=2:idle mode, *pIntModeValue=3:sleep mode
*pIntModeValue=-1:failed

- BOOL GetCompassStatusValue(INT32* pIntStatusValue):

Return : true if succeed, false if fail
Output : INT32* pIntStatusValue
*pIntStatusValue=1 : X,Y,Z data are ready to read.
*pIntStatusValue=2 : Not all for of the x,y,z data have been read
*pIntStatusValue=-1:failed

- BOOL GetCompassData(UINT32* pIntX, UINT32* pIntY, UINT32* pIntZ):

Return : true if succeed, false if fail
Output : UINT32* pIntX, UINT32* pIntY, UINT32* pIntZ
X,Y,Z axis integer data, range from 63488 to 2047(0xF800 to 0x07FF)

6. G-Sensor API:

- BOOL GetGSensorData(UINT32* pIntX, UINT32* pIntY, UINT32* pIntZ):

Return : true if succeed, false if fail
Output : UINT32* pIntX, UINT32* pIntY, UINT32* pIntZ
X,Y,Z axis integer data

OBDII API (CEIcpFleetMangDll.dll):

- ICPFM_API bool ICPFMDLLAPI ICPFM_Init(HWND hWnd, PORT_CFG mPort):

Description:
Initialize GPS/OBD port number and baudrate.
Input Parameters:
hWnd : The handle of MFC program.
mPort : The GPS/OBD port number and badrate.
Return Value:
true : Success to initialize the configuration.
false : Failed to initialize the configuration.
Example :

```
PORT_CFG mPortCfg;
HWND hWnd;
hWnd = this->m_hWnd;
mPortCfg.iGpsPort = 1;//or 2 or 1
mPortCfg.iObdPort = 3;
mPortCfg.dwGpsBaudrate = 4800;
mPortCfg.dwObdBaudrate = 115200;

ICPFM_Init( hWnd, mPortCfg );
```

- ICPFM_API bool ICPFMDLLAPI ICPFM_Start(void):

Description:
Start to detect GPS/OBD and alert status.
Input Parameters:
None
Return Value:
true : Success to start operation.

false : Failed to start operation.
Example :
ICPFM_Start();

- **ICPFM_API bool ICPFMDLLAPI ICPFM_GetObd(TCHAR tObd[][64]):**

Description:
Query the newest OBD data
Input Parameters:
pVersion : The pointer to a buffer that store OBD status
Return Value:
true : Success to query OBD info.
false : Failed to query OBD info.
Example :
TCHAR tObd[100][64];
ICPFM_GetObd(tObd)

- **ICPFM_API bool ICPFMDLLAPI GetObdII(int iIndex, TCHAR *ptszBuffer):**

Description:
Get OBDII data from DLL.
Input Parameters:
iIndex : The PID of OBDII.
ptszBuffer : The buffer point that store corresponding value of PID.
ABOUT PID please referecne WIKI OBDII
http://en.wikipedia.org/wiki/OBD-II_PIDs

Return Value:
true : The ECU of vehicle provide such a data
false : The ECU of vehicle doesn't provide such a data
Example :
bool bRtn;
int idx;
TCHAR tsz[128];
bRtn = GetObdII(idx, tsz);