

# SSP Update 'C' File Implementation Version 1.0

INTELLIGENCE IN VALIDATION

## TABLE OF CONTENTS

Table of Contents.....	2
1 Introduction.....	3
2 Version History.....	3
3 File structure.....	4
4 Visual C++ 6 example implementation.....	6
5 Function handler implementation.....	8
5.1 OpenComPort.....	8
5.2 CloseComPort.....	9
5.3 WaitDelay.....	10
5.4 SendComData .....	11
5.5 GetTargetChecksum.....	12
5.6 SetDownloadSpeed .....	13
5.7 TransmitDataPacket .....	14

## 1 INTRODUCTION.

This document describes a set of C source files to assist in the implementation of the ITL SSP remote update function.

The supplied source files are supplied with empty function handlers to allow the user to tailor the system to his particular hardware and platform requirements.

An example windows command line downloader is supplied to show how this is integrated into a system.

## 2 VERSION HISTORY.

Version 1.0 – Initial document – 10<sup>th</sup> November 2009

### 3 FILE STRUCTURE.

There are six files supplied in the 'Source File' Folder:

- Coms.c and Coms.h - provide the empty function handlers to be implemented for your system
- CTargetUpdate.c and CTargetUpdate.h - provide the download functions for the system.
- CsspUpdate.h - contains system structure definitions
- DefTypes.h - type definitions

The Folder 'VC6 example implementation' contains a full Visual C++6 project for a command line downloading system.

## Global Structure Definitions

The system uses two global structures:

**SSP\_PACKET:**

```
typedef struct{
    uint8 command[255]; // a buffer to hold the SSP command data
    uint8 commandLength; // a byte to hold command data length
    uint8 txPacketData[255]; // a buffer to hold the packet data to transmit
    uint8 txPacketLength; // a byte to hold the length of data to transmit
    uint8 retry; // the number of retries required for this packet
    uint8 replyStatus; // a byte to hold the packet reply status
    uint8 synchbit; // a byte to track the ssp synchronisation bit
    uint8 checkStuff; // a byte to track byte stuffing
    uint8 rxPtr; // a byte to hold the current received data length
    uint8 rxBufferLength; // a byte to hold total length of packet to receive
    uint8 rxData[255]; // a buffer to hold the received data
    uint16 timeout; // value in milliseconds giving the time to wait for a reply
}SSP_PACKET;
```

**SSP\_DOWNLOAD:**

```
typedef struct{
    int8* szFileName; // a pointer to the full filename to download to target
    int16 comPort; // the serial com port number of the system
    int16 PortStatus; // a variable to hold current port status
    int8 szFirmwareVersion[20]; // holds the firmware version of connected target
    int8 szDatasetVersion[20]; // holds the dataset version of connected target
    uint8* fileData; // a pointer to the loaded file data to download
}SSP_DOWNLOAD;
```

## 4 VISUAL C++ 6 EXAMPLE IMPLEMENTATION.

This example is a Visual C++ 6 project, which uses a simple command line downloader to update an SSP target.

Please note that the SSP system files have been rename with extension .cpp for this compile.

The file CsspUpdate contains the main function, which takes the command line parameters for file download.

### Command line parameters

There are two valid parameters:

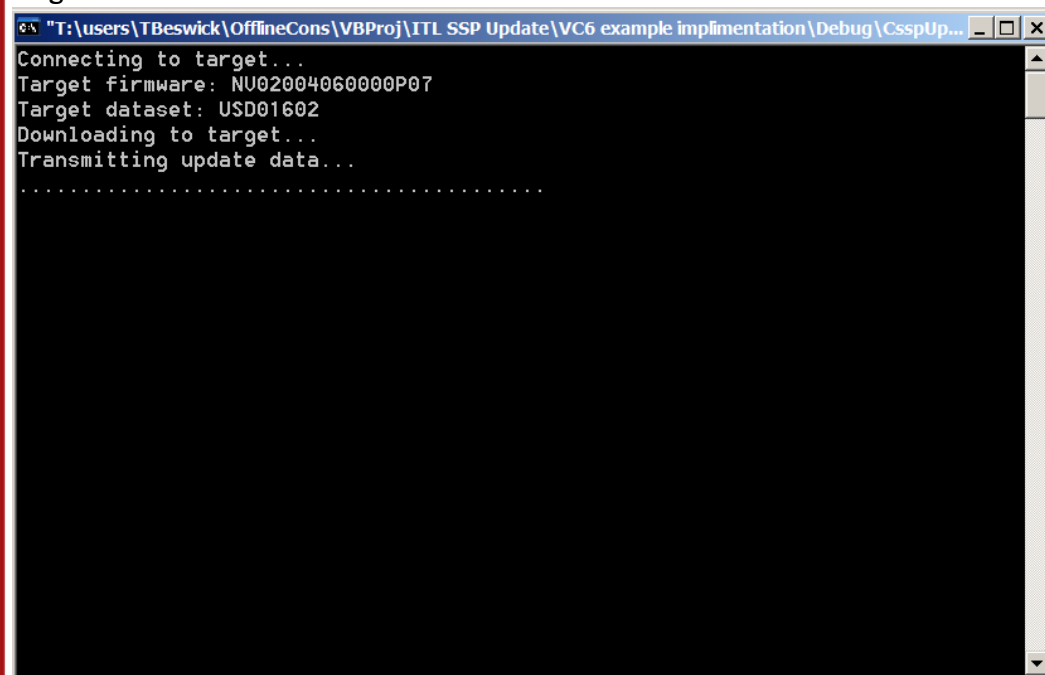
- -d        This is the full file path of the file to update
- -p        This is the serial communications port number

So an example command line would be

```
CsspUpdate -dC:\ITLSSPDownload\USD01602_NV02004060000P07_IF_01.bv1 -p7
```

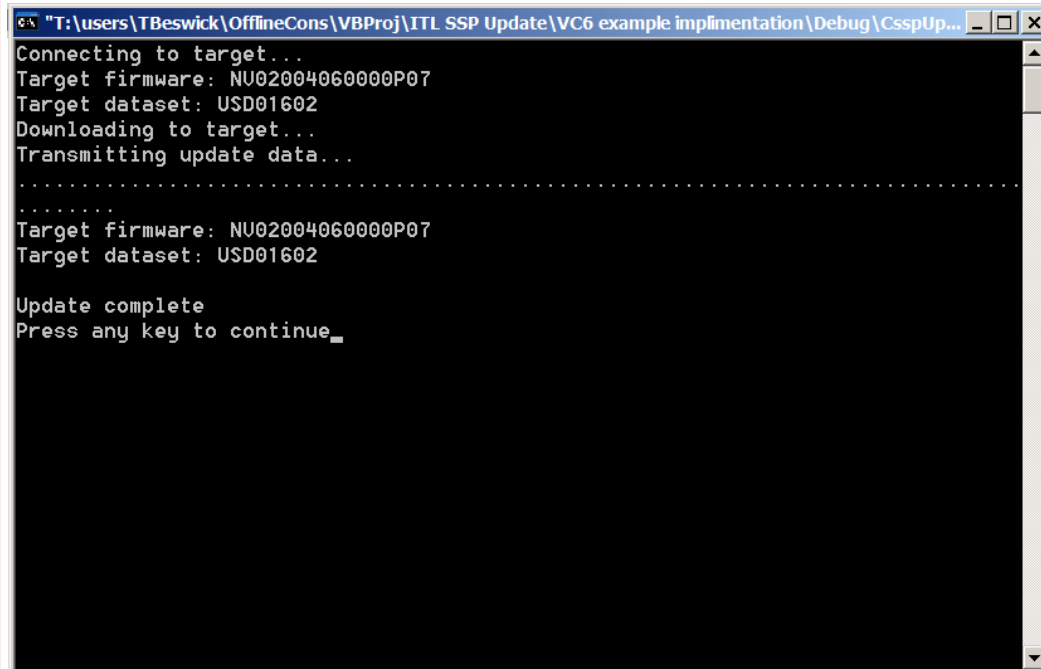
The software will test the command line parameters then check communications to the target and get the current dataset and firmware versions.

Target download will then commence.



```
T:\users\TBeswick\OfflineCons\VBProj\ITL SSP Update\VC6 example implimentation\Debug\CsspUp...
Connecting to target...
Target firmware: NU02004060000P07
Target dataset: USD01602
Downloading to target...
Transmitting update data...
.....
```

This is a completed update. The software checks the target version again at the end of the process.

A screenshot of a Windows command prompt window. The title bar shows the file path: "T:\users\TBeswick\OfflineCons\VBProj\ITL SSP Update\VC6 example implementation\Debug\CsspUp...". The window contains the following text:

```
Connecting to target...
Target firmware: NU02004060000P07
Target dataset: USD01602
Downloading to target...
Transmitting update data...
.....
Target firmware: NU02004060000P07
Target dataset: USD01602

Update complete
Press any key to continue_
```

## 5 FUNCTION HANDLER IMPLEMENTATION.

### 5.1 OPENCOMPORT

**Function Header:**

int16 OpenComPort(SSP\_DOWNLOAD\* sspD)

**Function description:**

This function is called to open a serial communication port on the user's system. In windows, this would use the CreateFile function. In an embedded system, this could be used to initialise a UART or USB port.

**Function Parameters:**

SSP\_DOWNLOAD\* sspD – a pointer to the Global SSP\_DOWNLOAD structure which must be pre-filled with the number of the serial com port to be opened.

**Function return:**

Returns 0 for port open failure, 1 for port open success.



## 5.2 CLOSECOMPORT

**Function Header:**

```
void CloseComPort(SSP_DOWNLOAD* sspD)
```

**Function description:**

This function is called to close a serial communication port. In Windows, this would use the CloseHandle function. In an embedded system, this may disable UART interrupts.

**Function Parameters:**

SSP\_DOWNLOAD\* sspD – a pointer to the Global SSP\_DOWNLOAD structure which must be pre-filled with the number of the serial com port to be opened.

**Function return:**

Returns 0 for port open failure, 1 for port open success.

### 5.3 WAITDELAY

**Function Header:**

```
void WaitDelay(uint32 delay)
```

**Function description:**

A function to provide a delay wait time for the system. In Windows, this may be the Sleep(delay) function in an embedded system a timer may be used to implement the delay

**Function Parameters:**

uint32 delay – a long parameter which is the required delay in milliseconds.

**Function return:**

None

## 5.4 SENDCOMDATA

**Function Header:**

int16 SendComData(uint8\* data, uint32 length)

**Function description:**

This function must send a serial data byte stream to the target. In Windows, this may use the WriteFile function. An embedded system would use the UART or USB to directly send this data buffer.

**Function Parameters:**

uint8\* data – a pointer to the data buffer to send.

uint32 length – the length of the data buffer to be sent

**Function return:**

0 for fail, 1 for success

## 5.5 GETTARGETCHECKSUM

**Function Header:**

int16 GetTargetChecksum(uint8 chk)

**Function description:**

This function waits for a received byte from the target and compares this to the passed a parameter. The function needs to include a time-out feature if no byte is received from the target within five seconds.

**Function Parameters:**

uint8 chk – the byte to compare against.

**Function return:**

0 for fail, 1 for success

## 5.6 SETDOWNLOADSPEED

**Function Header:**

int16 SetDownloadSpeed(SSP\_DOWNLOAD\* sspD,uint32 iBaud)

**Function description:**

A function to set the Baud rate parameters of the system UART. In Windows, this would use the GetCommState and SetCommState functions to change the rate. An embedded system can change the Baud rate register directly.

**Function Parameters:**

SSP\_DOWNLOAD\* sspD – pointer to the global download structure

uint32 iBaud – the rate to which to change to (e.g. 38400)

**Function return:**

0 for fail, 1 for success

## 5.7 TRANSMITDATAPACKET

**Function Header:**

int16 TransmitDataPacket(SSP\_PACKET\* sspP)

**Function description:**

A function to transmit the complied SSP packet and retry if no reply is received. The user is required to implement a timeout function in here depending on the hardware and platform used.

**Function Parameters:**

SSP\_PACKET\* sspP – pointer to the global SSP packet structure

**Function return:**

0 for fail, 1 for success