Zebra TC-21 QR Scanner Set-Up

Johnjimy K. Som

June 8, 2021

Documentation to set-up a Zebra TC21 to scan QR Barcodes in hexadecimal, store in a .csv or .xlsx file, send the files into an SMB network, and parse in decimal to determine item codes from that scanned QR code. Note: *Italicized Text* means it can be seen in the Zebra Device.

1 Zebra TC-21 Configurations

Requirements to scan QR code:

- Zebra TC21 [Model:TC210K], Android Version 10
- Scanning Application
- File Explorer Application
- Server Message Block (SMB) configuration skills

Data Wedge Application

Data Wedge is a built in application for Zebra devices, In order to have the scanning function to perform quickly and efficiently, the following changes needs to made:

- In the *Data Wedge* application, Profile:Profile0 (default) should be selected
- Scroll down to *Keystroke output* section, 'Inter character delay' is up to [10 ms]
- Scroll down and select *Basic Data formatting* section
- 'Send ENTER key' should be [✓], not unchecked

2 'Orca Scan' Android Package (APK)

Orca Scan is an application created by Cambridge App Lab Limited. *Orca Scan* will make the TC21 device: scan, export .csv, .xlsx to a file manager application that connects to an Server Message Block (SMB) service.

- In the *Data Wedge* application, Profile:Profile0 (default) should be selected
- Scroll down to *Data Wedge* section, 'Inter character delay' is up to [10 ms]
- Scroll down and select 'Basic Data formatting' section
- 'Send ENTER key' should be [✓], not unchecked

3 Decoding Bar-code

```
#include <iostream>
#include <string.h>
#include <string>
using namespace std;

string GetBinaryStringFromHexString(string);

int main(int argc, char *argv[]) // input is a 32
    character string

{
  unsigned int encodeVer, printArea, itemCode, packingDiv,
      productionYear, quantity, serialNumber, checkSum;
  string inHex, inBinary;


  for (int i = 1; i < argc; ++i)
  {
    inHex = argv[i];
    inBinary = GetBinaryStringFromHexString(inHex);
    //cout << inHex << "\n";
    //cout << inBinary << "\n";

  ...
    cout << "Serial Number: " << serialNumber << "\n";
    cout << "Checksum: " << checkSum << "\n";
  }

  return 0;
}

string GetBinaryStringFromHexString(string sHex)
{
  string sReturn = "";
  for (unsigned int i = 0; i < sHex.length(); ++i)
  {
    switch (toupper(sHex[i]))
    {
      case '0': sReturn.append("0000"); break;
      case '1': sReturn.append("0001"); break;
      case '2': sReturn.append("0010"); break;
      case '3': sReturn.append("0011"); break;
      case '4': sReturn.append("0100"); break;
      case '5': sReturn.append("0101"); break;
      case '6': sReturn.append("0110"); break;
      case '7': sReturn.append("0111"); break;
      case '8': sReturn.append("1000"); break;
      case '9': sReturn.append("1001"); break;
      case 'A': sReturn.append("1010"); break;
      case 'B': sReturn.append("1011"); break;
      case 'C': sReturn.append("1100"); break;
      case 'D': sReturn.append("1101"); break;
```

```
51          case 'E': sReturn.append("1110"); break;
52          case 'F': sReturn.append("1111"); break;
53        }
54      }
55      return sReturn;
56    }
```

Listing 1: Snippet of the C++ to decode the bar-code