# TECHNICAL CODE

## COMPRESSION TABLE OF SERVICE INFORMATION (SI) DESCRIPTIONS FOR DIGITAL TERRESTRIALTELEVISION BROADCAST SERVICE

**Developed by**

**Registered by**

**Issued date: 31 January 2013**

# DEVELOPMENT OF TECHNICAL CODES

The Communications and Multimedia Act 1998 ('the Act') provides for Technical Standards Forum designated under section 184 of the Act or the Malaysian Communications and Multimedia Commission ('the Commission') to prepare a technical code. The technical code prepared pursuant to section 185 of the Act shall consist of, at least, the requirement for network interoperability and the promotion of safety of network facilities.

Section 96 of the Act also provides for the Commission to determine a technical code in accordance with section 55 of the Act if the technical code is not developed under an applicable provision of the Act and it is unlikely to be developed by the Technical Standards Forum within a reasonable time.

In exercise of the power conferred by section 184 of the Act, the Commission has designated the Malaysian Technical Standards Forum Bhd ('MTSFB') as a Technical Standards Forum which is obligated, among others, to prepare the technical code under section 185 of the Act.

A technical code prepared in accordance with section 185 shall not be effective until it is registered by the Commission pursuant to section 95 of the Act.

For further information on the technical code, please contact:

**Malaysian Communications and Multimedia Commission (SKMM)**
Off Pesiaran Multimedia
63000 Cyberjaya
Selangor Darul Ehsan
MALAYSIA

Tel: +60 3 8688 8000
Fax: +60 3 8688 1000
http://www.skmm.gov.my

OR

**Malaysian Technical Standards Forum Bhd (MTSFB**)
L2-E-11, Lab 3, Digital Media Centre
Enterprise 4
Technology Park Malaysia
Lebuhraya Puchong –Sg Besi
Bukit Jalil
57000 Kuala Lumpur
MALAYSIA

Tel: +60 3 8996 5505/5509
Fax: +60 3 8996 5507
http://www.mtsfb.org.my

# CONTENTS



**Tables**

## Committee Representation

The Multimedia Terminal Working Group (MMT WG) under the Malaysian Technical Standards Forum Bhd (MTSFB) which developed this Technical Code consists of representatives from the following organisations:

Al Hijrah Media Corporation (TV Alhijrah)

Asiaspace Sdn. Bhd.

BVD Systems Sdn. Bhd.

Conax AS Asia Pacific

Digital TV Labs Limited

LG Electronics (M) Sdn. Bhd.

ManQana Sdn. Bhd.

Measat Broadcast Network Sdn. Bhd. (ASTRO)

Media Prima Berhad

Panasonic R&D Centre Malaysia Sdn. Bhd.

Radio Television Malaysia (RTM)

Rohde & Schwarz Malaysia Sdn. Bhd.

SIRIM Berhad

Sony EMCS (M) Sdn. Bhd.

Strategy and Technology

Telekom Malaysia Berhad

Telekom Research & Development Sdn. Bhd.

U Mobile Sdn. Bhd.

U Television Sdn. Bhd.

Y&R

# FOREWORD

This technical code for the Compression Table of Service Information (SI) Descriptions for Digital Terrestrial Television Broadcast Service ('this Technical Code') was developed pursuant to section 185 of the Act 588 by the Malaysian Technical Standard Forum Berhad ('MTSFB') via its Multimedia Terminal Working Group.

This Technical Code shall continue to be valid and effective until reviewed or cancelled.

# COMPRESSION TABLE OF SERVICE INFORMATION (SI) DESCRIPTIONS FOR DIGITAL TERRESTRIAL TELEVISION BROADCAST SERVICE

## Introduction

### Proposition

Huffman coding provides a language specific method for compression of long descriptions. It has been proven to provide up to 50% savings when used to compress long textual descriptions especially when related to transmission of EPG information.

This document does not outline the signaling or other DVB SI/PSI implementation specifics. It provides only the specification for actual compression table and sample implementation to aid understanding of the structure of the Huffman compression table.

### Exclusion

The sample code is provided purely for increasing the readability and understanding of the Huffman table provided here. It is not meant to be used in the final implementation of a receiver and it is provided without any warranty or guarantees either explicitly or implied in any way.

## 1. Scope

This document does not outline the signaling or other DVB SI/PSI implementation specifics. It provides only the specification for actual compression table and sample implementation to aid in understanding the structure of the Huffman compression table.

## 2. Glossary

| | |
|---|---|
| DTT | Digital Terrestrial Television |
| DVB | Digital Video Broadcast organization |
| DVB-CI | DVB-Common Interface |
| DVB-T | DVB-Terrestrial |
| DVB-T2 | DVB-Terrestrial T2 |
| EPG | Electronic Programme Guide |
| EIT | Event Information Table |
| FTA | Free to Air |
| HDCP | High-Bandwidth Digital Content Protection |
| HDMI | High-Definition Multimedia Interface |
| HDTV | High Definition Television |
| IRD | Integrated Receiver Decoder |
| IDTVs | Integrated Digital Televisions |
| May | Indicates an event or provision which is permitted, but not mandatory |
| MPEG | Moving Pictures Expert Group |
| Must | Indicates that a third party must comply to ensure correct operation |

| | |
|---|---|
| | (present tense) Indicates an existing provision |
| OSD | Onscreen Display |
| (Opt) | Optional |
| (Req) | Requirement |
| RF | Radio Frequency |
| SD | Standard Definition |
| SDTV | Standard Definition Television |
| SI | Service Information |
| S/PDIF | Sony/Philips Digital Interface |
| STB | Set-Top-Box, which is equivalent to a digital Terrestrial receiver |
| Shall | Indicates a mandatory provision |
| Should | Indicates a desirable, but not mandatory, provision |
| (TS) | Transport Stream: A data structure defined in ISO/IEC 13818-1 |
| UHF | Ultra-High Frequency |
| Will | Indicates an assumption about existing states or future events |

## 3. Huffman Compression Samples

The encoding and decoding samples provided in this section was developed using console application on Microsoft® Visual Studio Express 2010 (C++).

### 3.1 Encoding Sample

The algorithm in this section, attempts to encode the full text by splitting it into smaller phrases. The maximum size of each phrase is 4 characters. If the 4 characters cannot be found in the Huffman lookup table, a phrase of 3 characters is attempted next. So on and so forth.

If search is unsuccessful (raw character does not match any element in Huffman table), then it will be treated as escape character. In such event, *Encode* will add a special tag “eg: 1101” as an indicator and followed by escape character corresponding binary code.

Example: if character “®” is detected, resulted binary code would be “*1101*11010010”, where 11010010 (0xD2) is hexadecimal value of “®” in character code table 00 – Latin as specified in ETSI EN 300 468. *Encode* will only convert single character each time an escape character was detected.

```
// --------------- Main function for Encoding -----------------------------//
//   [input]   PhraseToEncode  (string)
//   [output]  CodedResult     (string)
//
//   successful   : coded string will be returned.
//   unsuccessful : ENCODE_FAIL string will be returned
//
//    The algorithm below, attempts to encode the full text by splitting
//  it to smaller phrases. The maximum size of the each phrase is 4
//  characters. If the 4 characters cannot be found in the huffman
//  lookup table, than a 3 character phrase will be attempted characters
//  is attempted. So on and so forth.
//-------------------------------------------------------------------------//

unsigned char *Encode(string TextToEncode,
                      struct HuffmanTree HuffmanTable[HUFFMAN_TABLE_MAX_LENGTH],
                     int *EncodedTextLength)
{
      // Encoding Progress position
      unsigned int ProgressPos = 0;

      // Binary code after a phrase has been matched
      string CodedTextBinary = "";

      // Complete Text encoded and stored as hex code
      string CodedTextHex = "";

      // Loop until encoding of the full text is finished
      while(ProgressPos < TextToEncode.length())
      {
            // IdxToHuffmanTable - Current search position within the Huffman Table
            int IdxToHuffmanTable = 0;

            // EscapeCharacterBinary - Temporary storage for escape character
            // To store binary code of single byte data only
            string EscapeCharacterBinary = "";

            // Continuously loop until matched phrase is located in Huffman Table
            // HuffmanTable[x].STRING = returns the string of entry (x)
            // HuffmanTable[x].CODE = returns the code for string from above

            // compare returns 0 when a match has been found
            bool notMatching = true;

            while(notMatching)
            {
                  // The compare function will compare the input string from position ProgressPos
                  // with the entry in the huffman table (entry IdxToHuffmanTable). Since the Table
                  // is ordered with the highest Priority on top, this function will search for a
                  // phrases that matches with 4 characters, then 3, 2 and finally 1


                  notMatching = TextToEncode.compare(ProgressPos,                 // Position in Input Text
                                (HuffmanTable[IdxToHuffmanTable].STRING).length(), // Length of Huffman Table Phrase
                                 HuffmanTable[IdxToHuffmanTable].STRING);         // Huffman Table Phrase

                  // If A Match was found
                  if(notMatching == false)
```

```
                {
                    // CodedTextBinary - Store encoded binary code
                    // Copy Binary code from matched characters to CodedTextBinary
                    // Not applicable to escape characters
                    CodedTextBinary.insert(CodedTextBinary.length(), HuffmanTable[IdxToHuffmanTable].CODE);

                    // Increase start position of character to encode
                    ProgressPos += (HuffmanTable[IdxToHuffmanTable].STRING).length();

                }
                else // Match Not Found
                {
                    IdxToHuffmanTable++;

                    //If Search was not successful, escape character is found!
                    if(HuffmanTable[IdxToHuffmanTable].STRING.compare(HUFFMAN_END_OF_TABLE_MARKER)==0)
                    {
                        // Escape code is 1 element after HUFFMAN_END_OF_TABLE_MARKER
                        IdxToHuffmanTable++;

                        // Insert Escape Code
                        CodedTextBinary.insert(CodedTextBinary.length(),
                                               HuffmanTable[IdxToHuffmanTable].CODE);

                        // Convert escape character from respective hexadecimal to binary
                        EscapeCharacterBinary = ConvertToBinaryString(TextToEncode, ProgressPos);

                        // Insert into binary string
                        CodedTextBinary.insert(CodedTextBinary.length(), EscapeCharacterBinary);

                        // Match case was found, set false to flag
                        notMatching = false;

                        // Increase start position by 1 character
                        ProgressPos += 1;
                    }
                }
            }
        }

        // Calculate encoded text length
        if( CodedTextBinary.length()%SIZE_OF_BYTE == 0 )
        {
            *EncodedTextLength = CodedTextBinary.length()/SIZE_OF_BYTE;
        }
        else
        {
            *EncodedTextLength = ( CodedTextBinary.length()/SIZE_OF_BYTE )+1;
        }

        // Once encoding is done,
        // Convert all binary data into hexadecimal character
        // HexCharacter        : Store all Encoded Hexadecimal data
        unsigned char *HexCharacter = ConvertToHexCharacter(CodedTextBinary);

        return HexCharacter;
    }
```

```
// ---- Function for Converting Binary to Hexadecimal Character --//
//  [input]    CodedPhrase
//  [output]   HexCodedResult
//
//    The algorithm below, attempts to convert binary code to
//  hexadecimal value and store as array of characters.
//-------------------------------------------------------------//

unsigned char *ConvertToHexCharacter(string BinaryCoded)
{
      // Current bit position
      unsigned int BitPos = 0;

      // Ensure that the Binary string is byte aligned by adding redundant bits of "1"
      while(BinaryCoded.length() % SIZE_OF_BYTE != 0)
      {
            BinaryCoded.insert(BinaryCoded.length(), "1");
      }

      // Numbers of characters converted into hexadecimal
      unsigned int CharPos = 0;

      // Allocate memory for unsigned char pointer
      unsigned char *HexCharacters = new unsigned char[(BinaryCoded.length()/8)];

      // BitPos - Current bit position in BinaryCoded
      while(BitPos < BinaryCoded.length())
      {
            char Character = 0x00;
            // Pos - Current bit position in each byte
            for(int Pos = SIZE_OF_BYTE-1; Pos >= 0; Pos--)
            {
                  if(BinaryCoded.at(BitPos) == '1')
                  {
                        Character += (0x01 << Pos);
                  }
                  BitPos++;
            }

            HexCharacters[CharPos] = Character;
            CharPos++;
      }

      return HexCharacters;
}


// ----- Function for Converting Hexadecimal to Binary -----//
//   [input]    HexadecimalText (string)
//   [input]    Pos             (int)    Position to do conversion
//   [output]   BinaryCode      (string)
//
//   The algorithm below, attempts to convert detected escape
// character to corresponding binary value.
//-------------------------------------------------------------//
string ConvertToBinaryString(string HexadecimalText, int Pos)
{
      // Storage for binary data character
      string BinaryCode = "";

      // Character to be convert from hexadecimal to binary
      char HexChar = HexadecimalText.at(Pos);

      for(int BitPos = SIZE_OF_BYTE-1; BitPos >= 0; BitPos--)
      {
            char BitValue = '0';
            HexChar & (1 << BitPos) ? BitValue = '1' : BitValue = '0';
            BinaryCode.insert(BinaryCode.length(), 1, BitValue);
      }
            return BinaryCode;
}
```

### 3.2 Decoding Sample

The algorithm attempts to decode input hexadecimal data by using Huffman Table. Redundant data added during encoding process will be filtered out. Huffman table used in decoding must have the same elements and code pair as in encoding.

```
////////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------//
//  [input]    TextToDecode
//  [output]   DecodedText
//
//  successful  : decoded string will be returned.
//  unsuccessful: DECODE_FAILED string will be returned
//
//    Algorithm below, attemps to decode input hexadecimal data
//  by using Huffman Table. Redundant data added during encoding
//  will be filter out
//--------------------------------------------------------------------//

// Reject all redundant data at the end of Encoded information
// Maximum only 7 bits of "0" will be added from Huffman Encoding

unsigned char * Decode  (unsigned char *TextToDecode,
                         struct HuffmanTree HuffmanTable[HUFFMAN_TABLE_MAX_LENGTH],
                         int EncodedTextLength)
{
        // Decoding Progress position
        unsigned int ProgressPos = 0;

        // Binary code converted from input data
        string CodedTextBinary = "";

        unsigned char *DecodedText = new unsigned char[DESCRIPTOR_MAX_LENGTH];

        // Complete text decoded
        int DecodeTextLength = 0;

        CodedTextBinary = ConvertToBinary(TextToDecode,EncodedTextLength);

        // Loop through the full string
        while(ProgressPos < CodedTextBinary.length())
        {
            // IdxToHuffmanTable - Current search position within the Huffman Table
            int IdxToHuffmanTable = 0;

            // EscapeCharacterBinary - Binary code of Escape Character detected
            string EscapeCharacterBinary = "";

            // RemainingPos - Remaining binary code to be decode
            unsigned int RemainingPos = 0;

            // Continuous loop until match character is located in Huffman Table
            // HuffmanTable[x].STRING = returns string of character
            // HuffmanTable[x].CODE = returns corresponding binary code
            bool notMatching = true;

            while(notMatching)
            {
                  notMatching = CodedTextBinary.compare(ProgressPos,         // Position in Binary Input Text
                                (HuffmanTable[IdxToHuffmanTable].CODE).length(), // Length of Huffman Table Binary Phrase
                                HuffmanTable[IdxToHuffmanTable].CODE );          // Huffman Table Binary Phrase

                  // A match case is detected
                  if(notMatching == false)
                  {
                     // Increase start position of character to decode
                     ProgressPos += (HuffmanTable[IdxToHuffmanTable].CODE).length();

                     // DecodedText - Store decoded text
                     for(int StringPos=0;
                           StringPos!=HuffmanTable[IdxToHuffmanTable].STRING.length();
                           StringPos++)
                     {
                        DecodedText[DecodeTextLength] = HuffmanTable[IdxToHuffmanTable].STRING[StringPos];
```

```
                        DecodeTextLength++;
                    }
                }
                else
                {
                    // Continue to search in next element of HuffmanTable
                    IdxToHuffmanTable++;

                    //If reach end of Huffman Table, Search was not successful!
                    if(HuffmanTable[IdxToHuffmanTable].STRING.compare(HUFFMAN_END_OF_TABLE_MARKER)==0)
                    {
                        //Escape character code is 1 element after HUFFMAN_END_OF_TABLE_MARKER
                        IdxToHuffmanTable++;

                        // Check for Escape Charactrer header
                        if(CodedTextBinary.compare(ProgressPos,                     // Position in Binary Input Text
                                (HuffmanTable[IdxToHuffmanTable].CODE).length(), //Length of Huffman Table Binary Phrase
                                 HuffmanTable[IdxToHuffmanTable].CODE)==0)       //Huffman Table Phrase
                        {
                            // Escape Character header is matched
                            notMatching = false;

                            // Increase progress position, ignore Escape code header
                            ProgressPos += HuffmanTable[IdxToHuffmanTable].CODE.length();

                            // Get binary code for escape character (1 Byte)
                            EscapeCharacterBinary.insert(0, CodedTextBinary, ProgressPos, SIZE_OF_BYTE);

]                           // Convert to hexadecimal value
                            // Insert escape character into DecodedText
                            DecodedText[DecodeTextLength] = ConvertBinarytoHex(EscapeCharacterBinary);
                            DecodeTextLength++;

                            // Increase progress position by 1 Byte
                            ProgressPos += SIZE_OF_BYTE;
                        }
                        else
                        {
                            return NULL;
                        }
                    }
                }
            }

            RemainingPos = CodedTextBinary.length() - ProgressPos;

            // Used to detect redundant data at end of Encoded data
            if((RemainingPos < SIZE_OF_BYTE) &&
                        (CodedTextBinary.compare(ProgressPos, RemainingPos, NULL_DATA, 0, RemainingPos)==0))
            {
                return DecodedText;
            }
        }

        return DecodedText;
]   }



//-------------------------------------------------------------//
//  Special case when Escape Character is detected
//  Convert binary code back to hexadecimal
//-------------------------------------------------------------//
char ConvertBinarytoHex(string CharacterInBinary)
{
    char CharacterInHex = 0x00;
    char BinaryValue = 0x01;

    for(int i = 0; i < SIZE_OF_BYTE; i++)
    {
        if(CharacterInBinary[i] == '1')
        {
            CharacterInHex |= (BinaryValue << (SIZE_OF_BYTE - i - 1));
        }
    }
    return CharacterInHex;
}
```

```
string ConvertToBinary(unsigned char *HexadecimalText,int HexadecimalTextLength)
{
	// Storage for binary data character
	string BinaryCode = "";

	for(int Pos = 0; Pos < HexadecimalTextLength; Pos++)
	{
		int HexChar = HexadecimalText[Pos];
		for(int BitPos = SIZE_OF_BYTE-1; BitPos >= 0; BitPos--)
		{
			char BitValue = '0';
			int test = (HexChar >> BitPos);
			test&=0x01;
			if(test==0x01)
				BitValue = '1';
			else
				BitValue='0';

			//HexChar & (1 << BitPos) ? BitValue = '1' : BitValue = '0';
			BinaryCode.insert(BinaryCode.length(), 1, BitValue);
		}
	}

	return BinaryCode;
}
```

### 3.3 Huffman Table Code

The huffman table itself can be found in section 4. The items below are used in the examples above the create the table.

```
#ifndef __HUFFMAN_TABLE_H
#define __HUFFMAN_TABLE_H

#include <fstream>
#include <string>

#define HUFFMAN_TABLE_MAX_LENGTH 1000
#define DESCRIPTOR_MAX_LENGTH 256

#define HUFFMAN_TABLE_STRING 0
#define HUFFMAN_TABLE_CODE   1
#define HUFFMAN_TABLE_ELEMENT_SIZE 2

#define SIZE_OF_BYTE 8
#define SIZE_OF_NIBBLE 4
#define HEX_TAG_LENGTH 2

#define HUFFMAN_END_OF_TABLE_MARKER "END_OF_TABLE"
#define HUFFMAN_ESCAPE_CHAR_MARKER "ESCAPE"
#define NULL_DATA "11111111"

struct HuffmanTree{
    std::string STRING;
    std::string CODE;
};
#endif
```

## 4. Huffman Table

The Huffman coding needs to be optimized for each language, as such in the Malaysian context, there are two tables defined, one each for English and Bahasa Melayu.

### 4.1 Bahasa Melayu Table

**Table 1: Bahasa Melayu Huffman Table**

| Characters | Binary Code |
|---|---|
| "di A" | 010101101 |
| "sika" | 010111001 |
| "ng m" | 011100010 |
| "aksi" | 011110011 |
| "ma. " | 101000100 |
| "ikan" | 101010111 |
| ". Sa" | 101100110 |
| "isah" | 110100110 |
| "nya " | 110111011 |
| " mem" | 111000010 |
| "gan " | 111001001 |
| " men" | 111001111 |
| " di " | 111111100 |
| "ngan" | 00100101 |
| " ber" | 01001000 |
| "dan " | 01001011 |
| " dan" | 01001101 |
| "kan " | 10100011 |
| " yan" | 0000011 |
| "yang" | 0001100 |
| "ang " | 1110110 |
| "eng" | 010111000 |
| "ya " | 010111010 |
| " be" | 011100011 |
| "a d" | 101000000 |
| "ak " | 101000101 |
| "a. " | 101100000 |
| "nya" | 101100111 |
| "di " | 110100111 |
| " se" | 110111100 |
| " ke" | 111000011 |
| " di" | 111001010 |
| "ah " | 111010000 |
| "dan" | 111111101 |
| " ya" | 00100110 |
| "yan" | 01001001 |
| "kan" | 01001100 |
| " da" | 01001110 |
| " me" | 10100100 |

| | |
|---|---|
| "ng " | 0000100 |
| "ang” | 0001101 |
| "an " | 1110111 |
| "hi" | 0101011000 |
| "ku" | 0101011001 |
| "ud" | 0110010010 |
| "um" | 0110010011 |
| " T" | 0111011001 |
| "lu" | 0111011010 |
| "id" | 0111011011 |
| " r" | 0111100100 |
| "Ma" | 1010000010 |
| "de" | 1010000011 |
| "us" | 1010000100 |
| " l" | 1010000101 |
| "mb" | 1010000110 |
| "rk" | 1010000111 |
| "Ha" | 1010010110 |
| "pu" | 1010010111 |
| "om" | 1010101000 |
| " h" | 1010101001 |
| "l " | 1010101010 |
| "ay" | 1010101011 |
| "As" | 1011000100 |
| " H" | 1011000101 |
| “ R" | 1011000110 |
| "i." | 1011000111 |
| "tr" | 1011001001 |
| "Pr" | 1101000110 |
| "on" | 1101000111 |
| "du" | 1101001000 |
| "ad" | 1101001001 |
| "ko" | 1101001010 |
| " K" | 1101001011 |
| "La" | 1101010101 |
| "le" | 1101010110 |
| "eb" | 1101010111 |
| " L" | 1101111011 |
| " c" | 1110000011 |
| "mp" | 1110010000 |
| "et" | 1110010001 |
| "ap" | 1110011000 |
| "ru" | 1110011001 |
| "ti" | 1110100010 |
| "re" | 1110100011 |
| "gi" | 1111011111 |
| "t " | 1111110000 |
| "st" | 1111110001 |
| "ag" | 1111110010 |

| | |
|---|---|
| "it" | 1111110011 |
| "tu" | 000000000 |
| "Sa" | 000000001 |
| "o " | 000001010 |
| "bu" | 000001011 |
| "u " | 000010100 |
| "ks" | 000010101 |
| "m " | 001000000 |
| "ro" | 001000001 |
| "ki" | 001000010 |
| "wa" | 001000100 |
| "ja" | 001000101 |
| "ia" | 010001000 |
| "il" | 010001001 |
| "ek" | 010011110 |
| "li" | 010111011 |
| "as" | 010111100 |
| "ni" | 010111101 |
| " a" | 011001000 |
| "pe” | 011101000 |
| " M" | 011101001 |
| " p" | 011101010 |
| "pa" | 101010110 |
| "im" | 101100001 |
| “a." | 101100101 |
| "te" | 110100000 |
| "is" | 110100001 |
| ", " | 110110000 |
| "un" | 110110001 |
| "nt" | 110110010 |
| "ny" | 110110011 |
| " t" | 110111010 |
| "ik" | 111000000 |
| "be" | 111000100 |
| "ua" | 111000101 |
| " S" | 111001011 |
| "ba" | 111001101 |
| " P" | 111001110 |
| "el" | 111010011 |
| "em" | 111101110 |
| "at" | 111111010 |
| "ai" | 111111011 |
| "si" | 00000001 |
| "ke" | 00000100 |
| "se" | 00001011 |
| "sa" | 00100011 |
| " y" | 00100100 |
| "ra" | 00100111 |
| " A" | 00111010 |

| | |
|---|---|
| "al" | 01000101 |
| "k " | 01001010 |
| "h " | 01010111 |
| "am" | 01011000 |
| "ha" | 01011001 |
| " b" | 01011111 |
| "na" | 01100101 |
| " s" | 01100110 |
| " k" | 01100111 |
| "in" | 01110000 |
| "di" | 01110111 |
| "ga" | 01111000 |
| "ar" | 10100110 |
| "ri" | 10100111 |
| "me" | 11010100 |
| "ta" | 11011010 |
| "en" | 11011011 |
| "ma" | 11011100 |
| " m" | 11011111 |
| "la" | 11101011 |
| "er" | 11110110 |
| "da" | 11111000 |
| "ak" | 11111001 |
| ". " | 11111111 |
| "ka" | 0000001 |
| "g " | 0001000 |
| "ah" | 0011001 |
| "ya" | 0100011 |
| " d" | 0101101 |
| "i " | 0111101 |
| "ng" | 1101011 |
| "a " | 1111010 |
| "n " | 000101 |
| "an" | 111100 |
| "'" | 010011111001 |
| "`" | 01001111101 |
| "{" | 01001111110 |
| "X" | 01001111111 |
| "<" | 01010000000 |
| ">" | 01010000001 |
| "}" | 01010000010 |
| "$" | 01010000011 |
| "#" | 01010000100 |
| "(" | 01010000101 |
| "_" | 01010000110 |
| "*" | 01010000111 |
| "@" | 01010001000 |
| "9" | 1010001001 |
| ")" | 01010001010 |

| | |
|---|---|
| "4" | 01010001011 |
| "6" | 01010001100 |
| "[" | 01010001101 |
| ";" | 01010001110 |
| ":" | 01010001111 |
| "^" | 01010010000 |
| "/", | 01010010001 |
| "\"" | 01010010010 |
| "x" | 01010010011 |
| "]" | 01010010100 |
| "\|" | 01010010101 |
| "%" | 01010010110 |
| "+" | 01010010111 |
| "~" | 01010011000 |
| "=" | 01010011001 |
| "U" | 01010011010 |
| "Q" | 01010011011 |
| "&" | 01010011100 |
| "5" | 01010011101 |
| "c" | 111000110 |
| "," | 111000111 |
| "S" | 111010100 |
| "P" | 111010101 |
| "A" | 00111011 |
| "." | 0001001 |
| "p" | 0011000 |
| "o" | 0011100 |
| "y" | 0111001 |
| "b" | 1010100 |
| "h" | 1111101 |
| "l" | 000011 |
| "d" | 000111 |
| "g" | 001101 |
| "t" | 001111 |
| "s" | 010000 |
| "u" | 011000 |
| "m" | 011111 |
| "r" | 101011 |
| "k" | 101101 |
| "e" | 00101 |
| "i" | 01101 |
| "n" | 10111 |
| " " | 1000 |
| "a" | 1100 |
| "v" | 01010011110 |
| "2" | 01010011111 |
| "G" | 01010100000 |
| "?" | 01010100001 |
| "8" | 01010100010 |

| | |
|---|---|
| "!" | 01010100011 |
| "O" | 01010100100 |
| "1" | 01010100101 |
| "0" | 01010100110 |
| "3" | 01010100111 |
| "Y" | 01010101000 |
| "7" | 01010101001 |
| "q" | 01010101010 |
| "V" | 01010101011 |
| "W" | 01010101100 |
| "Z" | 01010101101 |
| "E" | 01010101110 |
| "C" | 01010101111 |
| "J" | 01110110000 |
| "I" | 01110110001 |
| "F" | 11011110100 |
| "-" | 11011110101 |
| "N" | 11110111100 |
| "D" | 11110111101 |
| "B" | 0010000110 |
| "f" | 0010000111 |
| "T" | 0111100101 |
| "H" | 1011001000 |
| "K" | 1101010100 |
| "R" | 1110000010 |
| "L" | 1110100100 |
| "z" | 1110100101 |
| "M" | 011101011 |
| "w" | 101001010 |
| "j" | 110100010 |

### 4.2 English Table

**Table 2: English Huffman Table**

| | |
|---|---|
| “This" | 100010011 |
| "show" | 101100111 |
| " Thi" | 110000001 |
| " sho" | 110000010 |
| "s an" | 110000011 |
| "is s" | 110001011 |
| "for " | 110001100 |
| "tion" | 110010000 |
| ". Th" | 111110000 |
| " for" | 111111010 |
| "ith " | 00000101 |
| "with" | 00010100 |
| " wit" | 00010101 |

| | |
|---|---|
| " in " | 00101000 |
| " to " | 00101101 |
| "his " | 00110001 |
| " and" | 00110011 |
| " of " | 10110111 |
| "and " | 11001001 |
| "s a" | 100000111 |
| "in " | 100100100 |
| " wi" | 101110010 |
| "ng " | 110000100 |
| "his" | 110000101 |
| " to" | 110000110 |
| "to " | 110001101 |
| "s t" | 110001110 |
| "of " | 110010001 |
| " a " | 111110001 |
| " of" | 111111011 |
| "and" | 00000110 |
| "nd " | 00010110 |
| " an" | 00010111 |
| "ing" | 00101001 |
| " in" | 00110000 |
| "is " | 00110010 |
| "es " | 00111000 |
| "he " | 10111000 |
| "the" | 11001100 |
| "vi" | 1001001010 |
| "w " | 1001001011 |
| "fe" | 1001001100 |
| "ut" | 1001001101 |
| "mp" | 1001001110 |
| "mi" | 1001001111 |
| "ns" | 1001010000 |
| "sc" | 1001010001 |
| "ge" | 1010110010 |
| "ce" | 1010110011 |
| "rt" | 1010111100 |
| " r" | 1010111101 |
| "ol" | 1010111110 |
| "s." | 1010111111 |
| "em" | 1011100110 |
| "Th" | 1100001110 |
| "wh" | 1100001111 |
| "pl" | 1100010000 |
| "pr" | 1100010001 |
| "fa" | 1100010010 |
| "un" | 1100010011 |
| "ei" | 1100110101 |
| "ly" | 1100110110 |
| "ni" | 1100110111 |
| "so" | 1100111000 |
| " C" | 1100111001 |

| | |
|---|---|
| "el" | 000010000 |
| "us" | 000010001 |
| " e" | 000010010 |
| "rs" | 000010011 |
| "si" | 000010100 |
| "ir" | 000010101 |
| "ai" | 001001000 |
| "ll" | 001001001 |
| "na" | 001010100 |
| "ow" | 001010101 |
| "be" | 001110100 |
| "ur" | 001110101 |
| "om" | 001110110 |
| "ta" | 001110111 |
| "di" | 001111110 |
| "ha" | 001111111 |
| "ic" | 010011100 |
| "io" | 010011101 |
| "me" | 010111000 |
| "ra" | 010111001 |
| "sh" | 010111010 |
| " l" | 010111011 |
| "ts" | 010111100 |
| " T" | 010111101 |
| "ve" | 011110101 |
| "li" | 011110110 |
| "ro" | 011110111 |
| "se" | 011111000 |
| "la" | 100001010 |
| "ie" | 100001011 |
| "am" | 100010010 |
| "ca" | 100101011 |
| " d" | 101010100 |
| "ch" | 101010101 |
| "co" | 101011000 |
| "wi" | 101100100 |
| "nt" | 101100101 |
| "ne" | 101100110 |
| "fo" | 101110110 |
| "de" | 101110111 |
| "ed" | 110000000 |
| "g " | 110001010 |
| "ho" | 110001111 |
| "f " | 110011101 |
| “ou" | 110011110 |
| "as" | 110011111 |
| "l " | 111110010 |
| " p” | 111110011 |
| "ma" | 111111110 |
| "ea" | 111111111 |
| "h " | 00001011 |
| "ri" | 00001100 |

| | |
|---|---|
| " m" | 00001101 |
| "of" | 00001110 |
| "al" | 00001111 |
| "to" | 00100101 |
| "le" | 00101011 |
| "il" | 00101100 |
| "ti" | 00111001 |
| "a " | 00111100 |
| " h" | 00111101 |
| "o " | 00111110 |
| "y " | 01001111 |
| "st" | 01011000 |
| "te" | 01011001 |
| "t " | 01011010 |
| ", " | 01011111 |
| " c" | 01100000 |
| " b" | 01100001 |
| "or" | 01100010 |
| "ar" | 01100011 |
| " f" | 01111000 |
| "it" | 01111001 |
| "at" | 10000100 |
| "on" | 10001000 |
| ". ", | 10001011 |
| "ng" | 10010000 |
| "en" | 10010001 |
| "hi" | 10101011 |
| " o" | 10101101 |
| " i" | 10101110 |
| "r " | 10110100 |
| "is" | 10110101 |
| " w" | 10110110 |
| "re" | 10111100 |
| "nd" | 10111101 |
| "d " | 11111010 |
| "er" | 11111011 |
| " s" | 11111100 |
| "es" | 0000001 |
| {"n " | 0010011 |
| "an" | 0010111 |
| "he" | 0110010 |
| " a" | 1000011 |
| "th" | 1010100 |
| "in" | 1011000 |
| {" t" | 1110010 |
| "e " | 000100 |
| "s " | 010010 |
| "'" | 01111100100 |
| "%" | 01111100101 |
| "$" | 01111100110 |
| "{" | 01111100111 |
| ";" | 01111101000 |

| | |
|---|---|
| "^" | 01111101001 |
| "_" | 01111101010 |
| "<" | 01111101011 |
| {"4" | 01111101100 |
| "`" | 01111101101 |
| "Q" | 01111101110 |
| "#" | 01111101111 |
| "O" | 01111110000 |
| "@" | 01111110001 |
| ")" | 01111110010 |
| ":" | 01111110011 |
| "/" | 01111110100 |
| "?" | 01111110101 |
| "(" | 01111110110 |
| "]" | 01111110111 |
| "[" | 01111111000 |
| ">" | 01111111001 |
| "\|" | 01111111010 |
| "}" | 01111111011 |
| "8" | 01111111100 |
| "7" | 01111111101 |
| "*" | 01111111110 |
| "+" | 01111111111 |
| "~" | 10000000000 |
| "=" | 10000000001 |
| "X" | 10000000010 |
| "-" | 1001010010 |
| "C" | 000001000 |
| "T" | 011110100 |
| "k" | 00000111 |
| "," | 01011011 |
| "v" | 10001010 |
| "." | 10111010 |
| "y" | 11111110 |
| "b" | 0000000 |
| "w" | 0100110 |
| "p" | 0110011 |
| "g" | 1001011 |
| "f" | 1011111 |
| "u" | 1100101 |
| "m" | 1110011 |
| "c" | 001000 |
| "d" | 001101 |
| "l" | 100011 |
| "h" | 111000 |
| "r" | 00011 |
| "n" | 01000 |
| "o" | 01010 |
| "s" | 01101 |
| "t" | 01110 |
| "i" | 10011 |
| "a" | 10100 |

| | |
|---|---|
| "e" | 11101 |
| " " | 1101 |
| "U" | 10000000011 |
| "!" | 10000000100 |
| "5" | 10000000101 |
| "3" | 10000000110 |
| "V" | 10000000111 |
| "0" | 10000001000 |
| "Z" | 10000001001 |
| "Y" | 10000001010 |
| "6" | 10000001011 |
| "N" | 10000001100 |
| "9" | 10000001101 |
| "E" | 10000001110 |
| "D" | 10000001111 |
| "2" | 10000010000 |
| "L" | 10000010001 |
| "\"" | 10000010010 |
| "F" | 10000010011 |
| "z" | 10000010100 |
| "R" | 10000010101 |
| "J" | 10000010110 |
| "q" | 10000010111 |
| "G" | 10000011000 |
| "W" | 10000011001 |
| "K" | 10000011010 |
| "I" | 10000011011 |
| "H" | 10010101010 |
| "&" | 10010101011 |
| "x" | 10111001110 |
| "M" | 10111001111 |
| "1" | 11001101000 |
| "B" | 11001101001 |
| "P" | 0000010010 |
| "j" | 0000010011 |
| "S" | 1001010011 |
| "A" | 1001010100 |

## 5. Examples

In this section various examples are provided as a reference for basic operational checking.

### 5.1 Basic Example

Two examples are provided here to illustrate the basic operation of the encoding.

**Table 3: Example 1**

| | |
|---|---|
| Uncompressed string | *Ini adalah rentetan untuk menunjukkan algoritma yang digunakan di Huffman Malaysia* |
| UTF-8 bytes for uncompressed string (hexadecimal) | 49 6e 69 20 61 64 61 6c 61 68 20 72 65 6e 74 65 74 61 6e 20 75 6e 74 75 6b 20 6d 65 6e 75 6e 6a 75 6b 6b 61 6e 20 61 6c 67 6f 72 69 74 6d 61 20 79 61 6e 67 20 64 69 67 75 6e 61 6b 61 6e 20 64 69 20 48 75 66 66 6d 61 6e 20 4d 61 6c 61 79 73 69 61 |
| Bytes for string compressed with Huffman Table Bahasa Melayu | 76 2b d6 47 c7 5a bf 47 b2 e4 7b ec 40 09 5a 97 d8 e8 98 b6 8d 14 d3 94 e7 ee 03 10 ee 6e c7 e7 be 9e c8 60 87 21 f7 05 a0 ba dc 80 e7 |
| Bytes for string compressed with Huffman Table English | 83 79 bc 33 43 e8 17 79 65 e9 dc 9e 26 ec a0 e1 b2 38 98 27 94 1c 1c bc 38 e5 b1 3c ff b7 f8 be 28 fd 2f 13 a0 39 7a a4 f6 55 65 bf 7f fc 4e e7 87 d3 f8 29 4f |

**Table 4: Example 2**

| | |
|---|---|
| Uncompressed string | *Misi Advanger adalah melindungi Precious daripada jatuh ke dalam tangan Negative Syndicate yang ingin menggunakannya untuk tujuan jahat.* |
| UTF-8 bytes for uncompressed string (hexadecimal) | 4d 69 73 69 20 41 64 76 61 6e 67 65 72 20 61 64 61 6c 61 68 20 6d 65 6c 69 6e 64 75 6e 67 69 20 50 72 65 63 69 6f 75 73 20 64 61 72 69 70 61 64 61 20 6a 61 74 75 68 20 6b 65 20 64 61 6c 61 6d 20 74 61 6e 67 61 6e 20 4e 65 67 61 74 69 76 65 20 53 79 6e 64 69 63 61 74 65 20 79 61 6e 67 20 69 6e 67 69 6e 20 6d 65 6e 67 67 75 6e 61 6b 61 6e 6e 79 61 20 75 6e 74 75 6b 20 74 75 6a 75 61 6e 20 6a 61 68 61 74 2e |
| Bytes for string compressed with Huffman Table Bahasa Melayu | 75 e8 5e 9d 8e a7 86 fb 32 3e 3a d5 f5 17 77 d2 35 bd d1 8b c6 69 ca 11 3a 9e ad f1 04 50 02 b8 22 77 59 03 68 96 3d e1 5e 3a 25 3c 5e 5b 9b bb f1 bf 45 06 21 c3 df 17 53 59 bb 1f 9f 37 7b 10 02 50 03 45 c5 14 8a b2 78 9f |
| Bytes for string compressed with Huffman Table English | b9 f6 b3 d9 50 d8 a2 f5 95 a5 0d 0f a0 5a e1 ed 7b 89 cb 9e 82 57 84 27 42 35 46 39 b3 a1 a7 80 9c 26 50 b0 71 0d 0f 89 72 2f 2c be c0 ce cb 84 9b d7 65 3f eb d4 e4 20 9f c5 f1 45 32 45 c4 84 bc 4e 80 e5 d1 fc 79 89 bb 28 3f 2c a0 9e 52 fa 09 d0 fe eb af |

### 5.2 ESC Code Example

The following example, demonstrates the usage of the ESC characters.

**Table 5: Example with ESC Code**

| | |
|---|---|
| Uncompressed string | *RM10 adalah bersamaan dengan £2.05 atau ¥278.34* |
| UTF-8 bytes for uncompressed string (hexadecimal) | 52 4d 31 30 20 61 64 61 6c 61 68 20 62 65 72 73 61 6d 61 61 6e 20 64 65 6e 67 61 6e 20 a3 32 2e 30 35 20 61 74 61 75 20 a5 32 37 38 2e 33 34 |
| Bytes for string compressed with Huffman Table Bahasa Melayu | e0 9d 6a 95 53 32 3e 3a d5 f8 95 91 ee 77 a0 c9 62 68 d4 f8 95 4c a7 59 1b 41 49 a5 53 ea a5 51 09 54 ea 2f |
| Bytes for string compressed with Huffman Table English | 82 b7 3f 34 40 88 66 87 d0 2c e8 13 89 50 bd 52 46 59 7d f5 1c 10 ba 81 10 16 19 df 2e fa 96 08 3f d7 f9 75 01 9f 67 |

## Acknowledgement

**Members of Multimedia Terminal Working Group:**

| | |
|---|---|
| Jaafar Hj Mohamad Abu Bakar (Chairman) | Telekom Research & Development Sdn. Bhd. |
| Dr. Rohmad Fakeh (Vice Chairman) | Radio Television Malaysia (RTM) |
| Razaini Mohd Razali (Secretary) | SIRIM Berhad |
| Abdullah Shahadan | Asiaspace Sdn. Bhd. |
| R. Kamalakaran | BVD Systems Sdn. Bhd. |
| Stephen Lee | Conax AS Asia Pacific |
| Omar Giri Valliappan | Digital TV Labs Limited |
| Hyek Seong Kweon | LG Electronics (M) Sdn. Bhd. |
| Kim Hong Soo | LG Electronics (M) Sdn. Bhd. |
| Mohd Redzwan Yahya | ManQana Sdn. Bhd. |
| Mustafa Kamal Mamat | Measat Broadcast Network Sdn. Bhd. (ASTRO) |
| Dr. Ahmad Zaki Mohd Salleh | Media Prima Berhad |
| Sukumar Lechimanan | Media Prima Berhad |
| Tan Kwong Meng | Media Prima Berhad |
| Lee Cheng Fei | Panasonic R&D Centre Malaysia Sdn. Bhd. |
| Ngo Chuan Hai | Panasonic R&D Centre Malaysia Sdn. Bhd. |
| Tan Jek Thoon | Panasonic R&D Centre Malaysia Sdn. Bhd. |
| William Wong | Panasonic R&D Centre Malaysia Sdn. Bhd. |
| Magli Alias | Radio Television Malaysia (RTM) |
| Iza Halilia Abd. Halim | Rohde & Schwarz Malaysia Sdn. Bhd. |
| Chia Beng Riang | Sony EMCS (M) Sdn. Bhd. |
| Chia Beng Riang | Sony EMCS (M) Sdn. Bhd. |
| Mohd Aizul Mohd Tallaha | Sony EMCS (M) Sdn. Bhd. |
| Muzaffar Fakhruddin | Sony EMCS (M) Sdn. Bhd. |
| Norihisa Ina | Sony EMCS (M) Sdn. Bhd. |
| Rajeshwari Konesin | Sony EMCS (M) Sdn. Bhd. |
| Colin Prior | Strategy and Technology |
| Delina Shamsudin | U Mobile Sdn. Bhd. |
| Hamzah Burok | U Mobile Sdn. Bhd. |
| Ahmad Shahrim Abd Rashid | U Television Sdn. Bhd. |
| Fauzi Osman | U Television Sdn. Bhd. |