ISO/IEC JTC 1/SC 17
Cards and personal identification
Secretariat: BSI (United Kingdom)

Document type: Text for CD ballot or comment

Title: Notification of Ballot: ISO/IEC 10373-6:2011/PDAM 4 - Identification cards - Test methods - Part 6: Proximity cards - AMENDMENT 4 Bit rates higher than fc/16 and up to fc/2

Status:

WORK ITEM: 60116

STATUS: This ballot has been posted to the ISO Electronic balloting application and is available under the Balloting Portal, Committee Internal Balloting.

Date of document: 2011-07-14

Expected action: VOTE

Action due date: 2011-09-15

Email of secretary: chris.starr@ukpayments.org.uk

Committee URL: http://isotc.iso.org/livelink/livelink/open/jtc1sc17

ISO/IEC JTC 1/SC 17/WG 8 N 1814

Date: 2011-04-01

ISO/IEC 10373-6:2011/PDAM 4

ISO/IEC JTC 1/SC 17/WG 8

Secretariat: DIN

Identification cards — Test methods — Part 6: Proximity cards

AMENDMENT 4
Bit rates higher than *fc*/16 and up to *fc/2*

Cartes d'identification — Méthodes d'essai — Partie 6: Cartes de proximité

AMENDEMENT 4
Débits binaires supérieurs à fc/16 jusqu'à fc/2

Document type: International Standard
Document subtype: Amendment
Document stage: (30) Committee
Document language: E

C:\Userdata\Standards\10373-6\Amendment 4 VHBR\Editing Ispra 2011\Send to WG8 member review\wg8nxxxx_CD10373-6_AMD4_VHBR_ASK_V2.doc STD Version 2.1c2

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 4 to ISO/IEC 10373-6:2011 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and personal identification*.

Identification cards — Test methods — Part 6: Proximity cards

Amendment 4: Bit rates higher than *fc*/16 and up to *fc/2*

Page 10 of ISO/IEC 10373-6:2011, Table 2

Add the following table note for component C3:

"[d] Capacitor C3 shall be removed for testing bit rates higher than *fc*/16 and up to *fc*/2."

Page 34 of ISO/IEC 10373-6:2011

Add new sub clause E.2.1 and move existing paragraph of E.2 and Figure E.2 to this sub clause:

"

E.2.1 Sampling for bit rates up to *fc*/16

"

Page 35 of ISO/IEC 10373-6:2011

Add new sub clause E.2.2:

"

E.2.2 Sampling for bit rates higher than *fc*/16 and up to *fc*/2

The time and voltage data of more than one modulation pulse, preferably a complete S(DESELECT) command with at least 20 carrier periods before the first and after the last modulation pulse containing short and long modulation pulses (see Figure E.3) shall be transferred to a suitable computer.

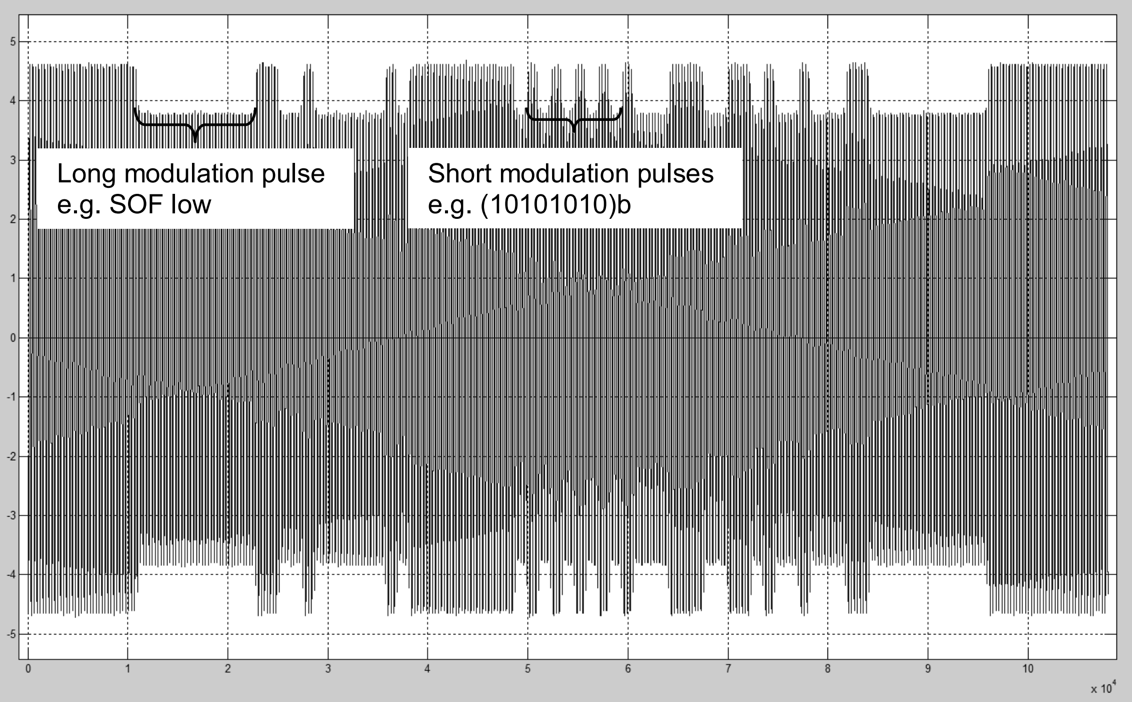

Figure E.3 — Modulation pulses

"

Page 35 of ISO/IEC 10373-6:2011

Add new sub clause E.3.1 and move existing paragraph of E.3 and Figure E.3 to this sub clause:

"

E.3.1 Filtering for bit rates up to *fc*/16

"

Page 35 of ISO/IEC 10373-6:2011

Add new sub clause E.3.2:

"

E.3.2 Filtering for bit rates higher than *fc*/16 and up to *fc*/2

A 4th order, Butterworth type band pass filter with center frequency of 13,56 MHz and 15 MHz 3-dB bandwidth shall be used for filtering the DC and higher harmonic components.

"

Page 36 of ISO/IEC 10373-6:2011

Add new sub clause E.5.1 and move existing paragraph of E.5 and Figure E.4 to this sub clause:

"

E.5.1 Envelope smoothing for bit rates up to *fc*/16

"

Page 36 of ISO/IEC 10373-6:2011

Add new sub clause E.5.2:

"

E.5.2 Envelope smoothing for bit rates higher than *fc*/16 and up to *fc*/2

No smoothing of signal envelope shall be applied.

"

Page 36 of ISO/IEC 10373-6:2011, E.6

Add at the end of paragraph:

"

For bit rates higher than *fc*/16 up to *fc*/2 and not settled envelope signals the minimum modulation index m_{min} shall be determined out of the part of the modulation sequence containing short modulation pulses (see Figure E.6).

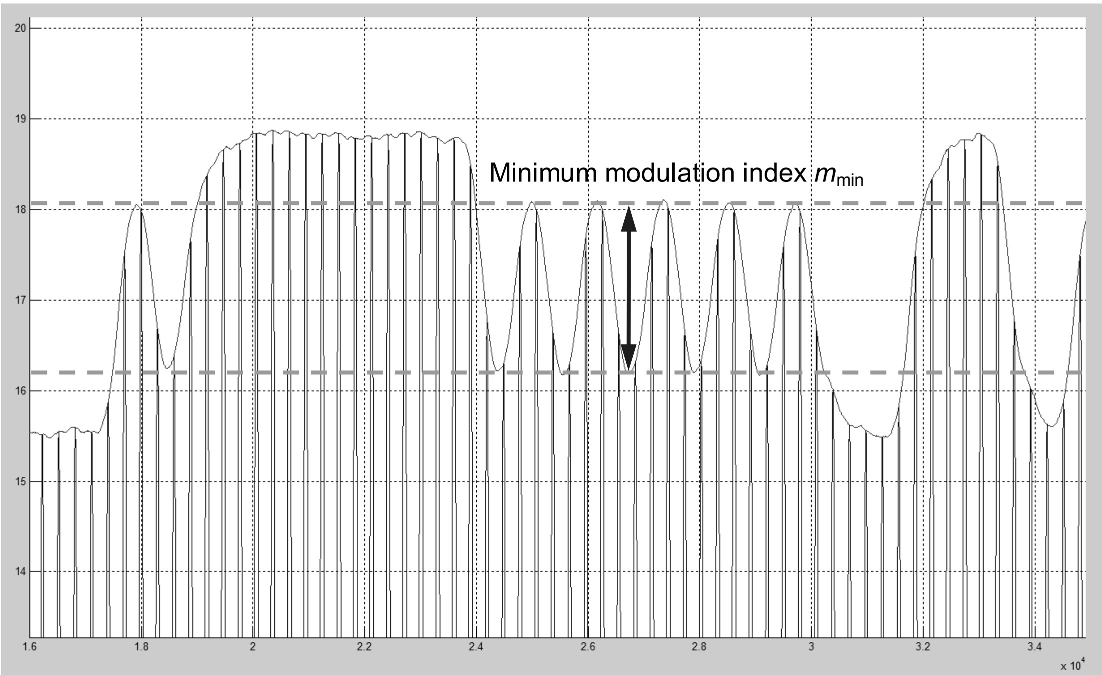

Figure E.6 — Minimum modulation index m_{min}

"

Page 36 of ISO/IEC 10373-6:2011, E.7

Add at the end of paragraph:

"

For bit rates higher than *fc*/16 up to *fc*/2 the timings shall be determined at positions with long modulation pulse positions e.g. t_f at transition to SOF low and t_r at transition to EOF high.

"

Page 80 of ISO/IEC 10373-6:2011

Replace Annex F with the following:

"

Annex F
(informative)

Program for the evaluation of the spectrum

"The following program written in C language gives an example for the calculation of the magnitude of the spectrum from the PICC.

```
/****************************************************************/
/***  This program calculates the Fourier coefficients       ***/
/***  of load modulated voltage of a PICC according          ***/
/***  the ISO/IEC 10373-6 Test methods                       ***/
/***  The coefficients are calculated at the frequencies:    ***/
/***    Carrier:        Fcm (=13.5600 for 13.56 MHz)         ***/
/***    Upper sideband: Fcm + fs                             ***/
/***    Lower sideband: Fcm - fs                             ***/
/***  fs is the subcarrier frequency and its value is:       ***/
/***  Fcm/16 for bit rates up to fc/16, Fcm/8, Fcm/4 or      ***/
/***  Fcm/2 for bit rates up to fc/2                         ***/
/****************************************************************/
/*** Input:                                                  ***/
/***  File in CSV Format containing a table of two           ***/
/***  columns (time and test PCD output voltage vd, clause 7)***/
/***                                                         ***/
/***  data format of input-file:                             ***/
/***  ------------------------                               ***/
/***  - one data-point per line:                             ***/
/***    (time[seconds], sense-coil-voltage[volts])           ***/
/***  - contents in ASCII, no headers                        ***/
/***  - data-points shall be equidistant time                ***/
/***  - minimum sampling rate: 100 MSamples/second           ***/
/***  - modulation waveform centered                         ***/
/***    (max. tolerance: half of subcarrier cycle)           ***/
/***                                                         ***/
/***                                                         ***/
/***  example for spreadsheet file (start in next line):     ***/
/***   (time)      (voltage)                                 ***/
/***  3.00000e-06,1.00                                       ***/
/***  3.00200e-06,1.01                                       ***/
/***  .....                                                  ***/
/****************************************************************/
/*** RUN:                                                    ***/
/***  “exefilename” [filename1[.csv] SubcarrierCode ]    ***/
/****************************************************************/
/*** ISO/IEC 10373-6 DFT CALCULATION                         ***/
/*** Version history:                                       ***/
/*** JUL 2000, version 1.1: original published version       ***/
/*** APR 2008, version 2.0: add the Barlett window           ***/
/*** NOV 2008, version 2.1: published version with revision  ***/
/*** SEP 2010, version 3.0: support higher subcarrier freq.  ***/
/****************************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
```

```
#define MAX_SAMPLES 50000
#define MAX_POINTS 500
#define MAX_MOYENNE 200

double  pi;   /* pi=3.14.... */

/* Array for time and sense coil voltage vd */
double vtime[MAX_SAMPLES]; /* time array                        */
double vd[MAX_SAMPLES];    /* Array for different coil voltage */



/*************************************************************/
/***    Read CSV File  Function                           ***/
/***                                                      ***/
/***    Description:                                      ***/
/***    This function reads the table of time and sense coil ***/
/***    voltage from a File in CSV Format                 ***/
/***                                                      ***/
/***    Input: filename                                   ***/
/***                                                      ***/
/***    Return: Number of samples  (sample Count)         ***/
/***            0  if an error occurred                   ***/
/***                                                      ***/
/***    Displays Statistics:                              ***/
/***                                                      ***/
/***    Filename, SampleCount, Sample rate, Max/Min Voltage  ***/
/*************************************************************/

int readcsv(char* fname)
{
   double a,b;
   double max_vd,min_vd;
   int i;
   FILE     *sample_file;

   /************ Open File **********************************/
   if (!strchr(fname, '.'))   strcat(fname, ".csv");

   if ((sample_file = fopen(fname, "r"))== NULL)
      {
            printf("Cannot open input file %s.\n",fname);
            return 0;
      }
   /*******************************************************/
   /* Read CSV File                                       */
   /*******************************************************/
   max_vd=-1e-9F;
   min_vd=-max_vd;
   i=0;

   while (!feof(sample_file))
      {
      if (i>=MAX_SAMPLES)
        {
        printf("Warning: File truncated !!!\n");
        printf("To much samples in file %s\b\n",fname);
        break;
        }
      fscanf(sample_file,"%Lf,%Lf\n", &a, &b);
      vtime[i] = a;
```

```
      vd[i] = b;
      if (vd[i]>max_vd) max_vd=vd[i];
      if (vd[i]<min_vd) min_vd=vd[i];
      i++;
      }
   fclose(sample_file);

   /************ Displays Statistics  ************************/
   printf("\n*******************************************\n");

   printf("\nStatistics: \n");
   printf(" Filename    : %s\n",fname);
   printf(" Sample count: %d\n",i);
   printf(" Sample rate : %1.0f MHz\n",1e-6/(vtime[1]-vtime[0]));
   printf(" Max(vd)     : %4.0f mV\n",max_vd*1000);
   printf(" Min(vd)     : %4.0f mV\n",min_vd*1000);
   return i;
}/**************** End ReadCsv ***************/


/*****************************************************************/
/***    DFT : Discrete Fourier Transformation                 ***/
/*****************************************************************/
/***    Description:                                          ***/
/***    This function calculate the Fourier coefficient       ***/
/***                                                          ***/
/***    Input: Number of samples                              ***/
/***           Carrier divider of the subcarrier              ***/
/***                                                          ***/
/***    Global Variables:                                     ***/
/***                                                          ***/
/***    Displays Results:                                     ***/
/***                                                          ***/
/***     Carrier coefficient                                  ***/
/***     Upper sideband coefficient                           ***/
/***     Lower sideband coefficient                           ***/
/***                                                          ***/
/*****************************************************************/
void dft(int count, int CarrierDivider)
{
  double c0_real,c0_imag,c0_abs,c0_phase;
  double c1_real,c1_imag,c1_abs,c1_phase;
  double c2_real,c2_imag,c2_abs,c2_phase;
  int N_data,center,start;
  double w0,wu,wl;
  double Wb;        /* Bartlett window coefficient */

  int i,k;

  double fc;        /* add variable for carrier frequency */


  fc=13.56e6;

  w0=(double)(fc*2.0)*pi; /* carrier 13.56 MHz */
  wu=(double)(1.0+1.0/CarrierDivider)*w0; /* upper sideband 14.41 MHz */
  wl=(double)(1.0-1.0/CarrierDivider)*w0; /* lower sideband 12.71 MHz */
  c0_real=0; /* real part of the carrier fourier coefficient */
  c0_imag=0; /* imag part of the carrier fourier coefficient */
  c1_real=0; /* real part of the up. sideband fourier coefficient */
  c1_imag=0; /* imag part of the up. sideband fourier coefficient */
```

```
c2_real=0; /* real part of the lo. sideband fourier coefficient */
c2_imag=0; /* imag part of the lo. sideband fourier coefficient */

center=(count+1)/2; /* center address */

/********** signal selection *****************************/

/* Number of samples for six subcarrier periods */

N_data=(int)(0.5+6.0F*CarrierDivider/(vtime[2]-vtime[1])/fc);

/* Note: (vtime[2]-vtime[1]) is the scope sample rate */

start=center - (int) N_data / 2;

/****************** DFT *******************************/

for( i=0;i<=N_data-1;i++)
{
/* Bartlett window */
if ((N_data & 1) == 0)
{
  /* N_data is even */
  if (i < (int) N_data /2)
  {
    Wb=2.0F*i/(double)(N_data - 1);
  }
  else
  {
    Wb=2.0F*(N_data-i-1)/(double)(N_data - 1);
  }
}
else
{
  /*N_data is odd */
  if (i < (int) N_data /2)
  {
    Wb=2.0F*i/(double)(N_data - 1);
  }
  else
  {
    Wb=2.0F-2.0F*i/(double)(N_data - 1);
  }

}

k=i+start;

c0_real=c0_real+vd[k]*(double)cos(w0*vtime[k])*Wb;
c0_imag=c0_imag+vd[k]*(double)sin(w0*vtime[k])*Wb;
c1_real=c1_real+vd[k]*(double)cos(wu*vtime[k])*Wb;
c1_imag=c1_imag+vd[k]*(double)sin(wu*vtime[k])*Wb;
c2_real=c2_real+vd[k]*(double)cos(wl*vtime[k])*Wb;
c2_imag=c2_imag+vd[k]*(double)sin(wl*vtime[k])*Wb;
}

/****************** DFT scale **************************/

c0_real=4.0F*c0_real/(double) N_data;
c0_imag=4.0F*c0_imag/(double) N_data;
c1_real=4.0F*c1_real/(double) N_data;
```

```
  c1_imag=4.0F*c1_imag/(double) N_data;
  c2_real=4.0F*c2_real/(double) N_data;
  c2_imag=4.0F*c2_imag/(double) N_data;
  /* Note: 4.0F includes the correction coef. of the bartlett window */

  /**************** absolute fourier coefficient **********/
  c0_abs=(double)sqrt(c0_real*c0_real + c0_imag*c0_imag);
  c1_abs=(double)sqrt(c1_real*c1_real + c1_imag*c1_imag);
  c2_abs=(double)sqrt(c2_real*c2_real+c2_imag*c2_imag);

  /************** Phase of fourier coefficient **********/
  c0_phase=(double)atan2(c0_imag,c0_real);
  c1_phase=(double)atan2(c1_imag,c1_real);
  c2_phase=(double)atan2(c2_imag,c2_real);

  /************** Result Display *************************/

  printf("\n\nResults: \n");
  printf("Subcarrier frequency = fc/%d\n",CarrierDivider);
  printf("Carrier ");
  printf("Abs: %7.3fmV ",1000*c0_abs);
  printf("Phase: %3.0fdeg\n",c0_phase/pi*180);
  printf("Upper sideband ");
  printf("Abs: %7.3fmV ",1000*c1_abs);
  printf("Phase: %3.0fdeg\n",c1_phase/pi*180);
  printf("Lower sideband ");
  printf("Abs: %7.3fmV ",1000*c2_abs);
  printf("Phase: %3.0fdeg\n\n",c2_phase/pi*180);
  printf("\n*********************************************\n");
  return;
  }/**************** End DFT ***************/


/*****************************************************************/
/***    MAIN  Program                                         ***/
/*****************************************************************/
int main(unsigned short paramCount,char *paramList[])
{
   char fname[256];
   unsigned int  sample_count;
   int Dsi; /* Subcarrier frequency code */
   int Ds[4]={2, 4, 8, 16}; /* Carrier divider */

   pi = (double)atan(1.0)*4; /* calculate pi */


   printf("\n*********************************************\n");
   printf("\n**** ISO/IEC 10373-6 PICC Test-Program     ****\n");
   printf("\n**** Version: 3.0 SEPTEMBER 2010            ****\n");
   printf("\n****                    ****\n");
   printf("\n*********************************************\n");
   /************* No Input Parameter           *****************/
   if (paramCount==1)
   {
      printf("\nCSV File name :");
      scanf("%s",fname);
    printf("\nSubcarrier frequency code [1=fc/2, 2=fc/4, 3=fc/8 or 4=fc/16)] :");
      scanf("%d",&Dsi);
   }
   else
   {
```

```
      /************* Input Parameter Loop     *****************/
      strcpy(fname,paramList[1]);
    if (!strchr(fname, '.'))   strcat(fname, ".csv");
    if (paramCount>2)
    {
      Dsi=atoi(paramList[2]);
    }
    else
    {
      Dsi=4; /*** for backward compatibility ***/
    }
   }
   if (!strchr(fname, '.'))   strcat(fname, ".csv");
   if (!(sample_count=readcsv(fname))) return 0;
   if ((Dsi > 0) & (Dsi < 5))
   {
     dft(sample_count,Ds[Dsi-1]);
   }
   else
   {
     printf("\nError: Subcarrier frequency code shall be 1,2,3,or 4\n");
   }

   return 0;
}/*********************** End Main  **************************/
```

"