Table of Contents

Code-128 Generator

```
<?php

/*******************************************************************************
* Script :  PDF_Code128
* Version : 1.2
* Date :    2016-01-31
* Auteur :  Roland Gautier
*
* Version   Date        Detail
* 1.2       2016-01-31  Compatibility with FPDF 1.8
* 1.1       2015-04-10  128 control characters FNC1 to FNC4 accepted
* 1.0       2008-05-20  First release
*
* Code128($x, $y, $code, $w, $h)
*     $x,$y :     angle supérieur gauche du code à barre
*                 upper left corner of the barcode
*     $code :     le code à créer
*                 ascii text to convert to barcode
*     $w :        largeur hors tout du code dans l'unité courante
*                 (prévoir 5 à 15 mm de blanc à droite et à gauche)
*                 barcode total width (current unit)
*                 (keep 5 to 15 mm white on left and right sides)
*     $h :        hauteur hors tout du code dans l'unité courante
*                 barcode total height (current unit)
*
* Commutation des jeux ABC automatique et optimisée
* Automatic and optimized A/B/C sets selection and switching
*
*
*   128 barcode control characters
*   ASCII   Aset            Bset        [ne pas utiliser][do not use]
*   --------------------------
*   200     FNC3            FNC3
*   201     FNC2            FNC2
*   202     ShiftA          ShiftB
*   203     [SwitchToCset]  [SwitchToCset]
*   204     [SwitchToBset]  FNC4
*   205     FNC4            [SwitchToAset]
*   206     FNC1            FNC1
*******************************************************************************/

require('fpdf.php');

class PDF_Code128 extends FPDF {

protected $T128;                                        // Tableau des codes 128
protected $ABCset = "";                                 // jeu des caractères éligibles au
C128
protected $Aset = "";                                   // Set A du jeu des caractères
éligibles
protected $Bset = "";                                   // Set B du jeu des caractères
éligibles
protected $Cset = "";                                   // Set C du jeu des caractères
éligibles
protected $SetFrom;                                     // Convertisseur source des jeux
vers le tableau
protected $SetTo;                                       // Convertisseur destination des
jeux vers le tableau
protected $JStart = array("A"=>103, "B"=>104, "C"=>105); // Caractères de sélection de jeu au
```

```
début du C128
protected $JSwap = array("A"=>101, "B"=>100, "C"=>99);   // Caractères de changement de jeu

//_________________________________ Extension du constructeur _____________________________
function __construct($orientation='P', $unit='mm', $format='A4') {

    parent::__construct($orientation,$unit,$format);

    $this->T128[] = array(2, 1, 2, 2, 2, 2);           //0 : [ ]               // composition
des caractères
    $this->T128[] = array(2, 2, 2, 1, 2, 2);           //1 : [!]
    $this->T128[] = array(2, 2, 2, 2, 2, 1);           //2 : ["]
    $this->T128[] = array(1, 2, 1, 2, 2, 3);           //3 : [#]
    $this->T128[] = array(1, 2, 1, 3, 2, 2);           //4 : [$]
    $this->T128[] = array(1, 3, 1, 2, 2, 2);           //5 : [%]
    $this->T128[] = array(1, 2, 2, 2, 1, 3);           //6 : [&]
    $this->T128[] = array(1, 2, 2, 3, 1, 2);           //7 : [']
    $this->T128[] = array(1, 3, 2, 2, 1, 2);           //8 : [(]
    $this->T128[] = array(2, 2, 1, 2, 1, 3);           //9 : [)]
    $this->T128[] = array(2, 2, 1, 3, 1, 2);           //10 : [*]
    $this->T128[] = array(2, 3, 1, 2, 1, 2);           //11 : [+]
    $this->T128[] = array(1, 1, 2, 2, 3, 2);           //12 : [,]
    $this->T128[] = array(1, 2, 2, 1, 3, 2);           //13 : [-]
    $this->T128[] = array(1, 2, 2, 2, 3, 1);           //14 : [.]
    $this->T128[] = array(1, 1, 3, 2, 2, 2);           //15 : [/]
    $this->T128[] = array(1, 2, 3, 1, 2, 2);           //16 : [0]
    $this->T128[] = array(1, 2, 3, 2, 2, 1);           //17 : [1]
    $this->T128[] = array(2, 2, 3, 2, 1, 1);           //18 : [2]
    $this->T128[] = array(2, 2, 1, 1, 3, 2);           //19 : [3]
    $this->T128[] = array(2, 2, 1, 2, 3, 1);           //20 : [4]
    $this->T128[] = array(2, 1, 3, 2, 1, 2);           //21 : [5]
    $this->T128[] = array(2, 2, 3, 1, 1, 2);           //22 : [6]
    $this->T128[] = array(3, 1, 2, 1, 3, 1);           //23 : [7]
    $this->T128[] = array(3, 1, 1, 2, 2, 2);           //24 : [8]
    $this->T128[] = array(3, 2, 1, 1, 2, 2);           //25 : [9]
    $this->T128[] = array(3, 2, 1, 2, 2, 1);           //26 : [:]
    $this->T128[] = array(3, 1, 2, 2, 1, 2);           //27 : [;]
    $this->T128[] = array(3, 2, 2, 1, 1, 2);           //28 : [<]
    $this->T128[] = array(3, 2, 2, 2, 1, 1);           //29 : [=]
    $this->T128[] = array(2, 1, 2, 1, 2, 3);           //30 : [>]
    $this->T128[] = array(2, 1, 2, 3, 2, 1);           //31 : [?]
    $this->T128[] = array(2, 3, 2, 1, 2, 1);           //32 : [@]
    $this->T128[] = array(1, 1, 1, 3, 2, 3);           //33 : [A]
    $this->T128[] = array(1, 3, 1, 1, 2, 3);           //34 : [B]
    $this->T128[] = array(1, 3, 1, 3, 2, 1);           //35 : [C]
    $this->T128[] = array(1, 1, 2, 3, 1, 3);           //36 : [D]
    $this->T128[] = array(1, 3, 2, 1, 1, 3);           //37 : [E]
    $this->T128[] = array(1, 3, 2, 3, 1, 1);           //38 : [F]
    $this->T128[] = array(2, 1, 1, 3, 1, 3);           //39 : [G]
    $this->T128[] = array(2, 3, 1, 1, 1, 3);           //40 : [H]
    $this->T128[] = array(2, 3, 1, 3, 1, 1);           //41 : [I]
    $this->T128[] = array(1, 1, 2, 1, 3, 3);           //42 : [J]
    $this->T128[] = array(1, 1, 2, 3, 3, 1);           //43 : [K]
    $this->T128[] = array(1, 3, 2, 1, 3, 1);           //44 : [L]
    $this->T128[] = array(1, 1, 3, 1, 2, 3);           //45 : [M]
    $this->T128[] = array(1, 1, 3, 3, 2, 1);           //46 : [N]
    $this->T128[] = array(1, 3, 3, 1, 2, 1);           //47 : [O]
    $this->T128[] = array(3, 1, 3, 1, 2, 1);           //48 : [P]
    $this->T128[] = array(2, 1, 1, 3, 3, 1);           //49 : [Q]
    $this->T128[] = array(2, 3, 1, 1, 3, 1);           //50 : [R]
    $this->T128[] = array(2, 1, 3, 1, 1, 3);           //51 : [S]
    $this->T128[] = array(2, 1, 3, 3, 1, 1);           //52 : [T]
```

```
    $this->T128[] = array(2, 1, 3, 1, 3, 1);           //53 : [U]
    $this->T128[] = array(3, 1, 1, 1, 2, 3);           //54 : [V]
    $this->T128[] = array(3, 1, 1, 3, 2, 1);           //55 : [W]
    $this->T128[] = array(3, 3, 1, 1, 2, 1);           //56 : [X]
    $this->T128[] = array(3, 1, 2, 1, 1, 3);           //57 : [Y]
    $this->T128[] = array(3, 1, 2, 3, 1, 1);           //58 : [Z]
    $this->T128[] = array(3, 3, 2, 1, 1, 1);           //59 : [[]
    $this->T128[] = array(3, 1, 4, 1, 1, 1);           //60 : [\]
    $this->T128[] = array(2, 2, 1, 4, 1, 1);           //61 : []]
    $this->T128[] = array(4, 3, 1, 1, 1, 1);           //62 : [^]
    $this->T128[] = array(1, 1, 1, 2, 2, 4);           //63 : [_]
    $this->T128[] = array(1, 1, 1, 4, 2, 2);           //64 : [`]
    $this->T128[] = array(1, 2, 1, 1, 2, 4);           //65 : [a]
    $this->T128[] = array(1, 2, 1, 4, 2, 1);           //66 : [b]
    $this->T128[] = array(1, 4, 1, 1, 2, 2);           //67 : [c]
    $this->T128[] = array(1, 4, 1, 2, 2, 1);           //68 : [d]
    $this->T128[] = array(1, 1, 2, 2, 1, 4);           //69 : [e]
    $this->T128[] = array(1, 1, 2, 4, 1, 2);           //70 : [f]
    $this->T128[] = array(1, 2, 2, 1, 1, 4);           //71 : [g]
    $this->T128[] = array(1, 2, 2, 4, 1, 1);           //72 : [h]
    $this->T128[] = array(1, 4, 2, 1, 1, 2);           //73 : [i]
    $this->T128[] = array(1, 4, 2, 2, 1, 1);           //74 : [j]
    $this->T128[] = array(2, 4, 1, 2, 1, 1);           //75 : [k]
    $this->T128[] = array(2, 2, 1, 1, 1, 4);           //76 : [l]
    $this->T128[] = array(4, 1, 3, 1, 1, 1);           //77 : [m]
    $this->T128[] = array(2, 4, 1, 1, 1, 2);           //78 : [n]
    $this->T128[] = array(1, 3, 4, 1, 1, 1);           //79 : [o]
    $this->T128[] = array(1, 1, 1, 2, 4, 2);           //80 : [p]
    $this->T128[] = array(1, 2, 1, 1, 4, 2);           //81 : [q]
    $this->T128[] = array(1, 2, 1, 2, 4, 1);           //82 : [r]
    $this->T128[] = array(1, 1, 4, 2, 1, 2);           //83 : [s]
    $this->T128[] = array(1, 2, 4, 1, 1, 2);           //84 : [t]
    $this->T128[] = array(1, 2, 4, 2, 1, 1);           //85 : [u]
    $this->T128[] = array(4, 1, 1, 2, 1, 2);           //86 : [v]
    $this->T128[] = array(4, 2, 1, 1, 1, 2);           //87 : [w]
    $this->T128[] = array(4, 2, 1, 2, 1, 1);           //88 : [x]
    $this->T128[] = array(2, 1, 2, 1, 4, 1);           //89 : [y]
    $this->T128[] = array(2, 1, 4, 1, 2, 1);           //90 : [z]
    $this->T128[] = array(4, 1, 2, 1, 2, 1);           //91 : [{]
    $this->T128[] = array(1, 1, 1, 1, 4, 3);           //92 : [|]
    $this->T128[] = array(1, 1, 1, 3, 4, 1);           //93 : [}]
    $this->T128[] = array(1, 3, 1, 1, 4, 1);           //94 : [~]
    $this->T128[] = array(1, 1, 4, 1, 1, 3);           //95 : [DEL]
    $this->T128[] = array(1, 1, 4, 3, 1, 1);           //96 : [FNC3]
    $this->T128[] = array(4, 1, 1, 1, 1, 3);           //97 : [FNC2]
    $this->T128[] = array(4, 1, 1, 3, 1, 1);           //98 : [SHIFT]
    $this->T128[] = array(1, 1, 3, 1, 4, 1);           //99 : [Cswap]
    $this->T128[] = array(1, 1, 4, 1, 3, 1);           //100 : [Bswap]
    $this->T128[] = array(3, 1, 1, 1, 4, 1);           //101 : [Aswap]
    $this->T128[] = array(4, 1, 1, 1, 3, 1);           //102 : [FNC1]
    $this->T128[] = array(2, 1, 1, 4, 1, 2);           //103 : [Astart]
    $this->T128[] = array(2, 1, 1, 2, 1, 4);           //104 : [Bstart]
    $this->T128[] = array(2, 1, 1, 2, 3, 2);           //105 : [Cstart]
    $this->T128[] = array(2, 3, 3, 1, 1, 1);           //106 : [STOP]
    $this->T128[] = array(2, 1);                       //107 : [END BAR]

    for ($i = 32; $i <= 95; $i++) {                                            // jeux de
caractères
        $this->ABCset .= chr($i);
    }
    $this->Aset = $this->ABCset;
    $this->Bset = $this->ABCset;
```

```
    for ($i = 0; $i <= 31; $i++) {
        $this->ABCset .= chr($i);
        $this->Aset .= chr($i);
    }
    for ($i = 96; $i <= 127; $i++) {
        $this->ABCset .= chr($i);
        $this->Bset .= chr($i);
    }
    for ($i = 200; $i <= 210; $i++) {                                       // controle
128
        $this->ABCset .= chr($i);
        $this->Aset .= chr($i);
        $this->Bset .= chr($i);
    }
    $this->Cset="0123456789".chr(206);

    for ($i=0; $i<96; $i++) {                                               //
convertisseurs des jeux A & B
        @$this->SetFrom["A"] .= chr($i);
        @$this->SetFrom["B"] .= chr($i + 32);
        @$this->SetTo["A"] .= chr(($i < 32) ? $i+64 : $i-32);
        @$this->SetTo["B"] .= chr($i);
    }
    for ($i=96; $i<107; $i++) {                                             // contrôle
des jeux A & B
        @$this->SetFrom["A"] .= chr($i + 104);
        @$this->SetFrom["B"] .= chr($i + 104);
        @$this->SetTo["A"] .= chr($i);
        @$this->SetTo["B"] .= chr($i);
    }
}

//_________________ Fonction encodage et dessin du code 128 _____________________
function Code128($x, $y, $code, $w, $h) {
    $Aguid = "";                                                                  //
Création des guides de choix ABC
    $Bguid = "";
    $Cguid = "";
    for ($i=0; $i < strlen($code); $i++) {
        $needle = substr($code,$i,1);
        $Aguid .= ((strpos($this->Aset,$needle)===false) ? "N" : "O");
        $Bguid .= ((strpos($this->Bset,$needle)===false) ? "N" : "O");
        $Cguid .= ((strpos($this->Cset,$needle)===false) ? "N" : "O");
    }

    $SminiC = "OOOO";
    $IminiC = 4;

    $crypt = "";
    while ($code > "") {
                                                                                  // BOUCLE
PRINCIPALE DE CODAGE
        $i = strpos($Cguid,$SminiC);                                              //
forçage du jeu C, si possible
        if ($i!==false) {
            $Aguid [$i] = "N";
            $Bguid [$i] = "N";
        }

        if (substr($Cguid,0,$IminiC) == $SminiC) {                                // jeu C
            $crypt .= chr(($crypt > "") ? $this->JSwap["C"] : $this->JStart["C"]);  // début
Cstart, sinon Cswap
```

```
            $made = strpos($Cguid,"N");                                          // étendu
du set C
            if ($made === false) {
                $made = strlen($Cguid);
            }
            if (fmod($made,2)==1) {
                $made--;                                                         //
seulement un nombre pair
            }
            for ($i=0; $i < $made; $i += 2) {
                $crypt .= chr(strval(substr($code,$i,2)));                       //
conversion 2 par 2
            }
            $jeu = "C";
        } else {
            $madeA = strpos($Aguid,"N");                                         // étendu
du set A
            if ($madeA === false) {
                $madeA = strlen($Aguid);
            }
            $madeB = strpos($Bguid,"N");                                         // étendu
du set B
            if ($madeB === false) {
                $madeB = strlen($Bguid);
            }
            $made = (($madeA < $madeB) ? $madeB : $madeA );                      // étendu
traitée
            $jeu = (($madeA < $madeB) ? "B" : "A" );                             // Jeu en
cours

            $crypt .= chr(($crypt > "") ? $this->JSwap[$jeu] : $this->JStart[$jeu]); // début
start, sinon swap

            $crypt .= strtr(substr($code, 0,$made), $this->SetFrom[$jeu],
$this->SetTo[$jeu]); // conversion selon jeu

        }
        $code = substr($code,$made);                                             // raccourcir
légende et guides de la zone traitée
        $Aguid = substr($Aguid,$made);
        $Bguid = substr($Bguid,$made);
        $Cguid = substr($Cguid,$made);
    }                                                                            // FIN BOUCLE
PRINCIPALE

    $check = ord($crypt[0]);                                                     // calcul de
la somme de contrôle
    for ($i=0; $i<strlen($crypt); $i++) {
        $check += (ord($crypt[$i]) * $i);
    }
    $check %= 103;

    $crypt .= chr($check) . chr(106) . chr(107);                                 // Chaine
cryptée complète

    $i = (strlen($crypt) * 11) - 8;                                              // calcul de
la largeur du module
    $modul = $w/$i;

    for ($i=0; $i<strlen($crypt); $i++) {                                        // BOUCLE
D'IMPRESSION
        $c = $this->T128[ord($crypt[$i])];
```

```
        for ($j=0; $j<count($c); $j++) {
            $this->Rect($x,$y,$c[$j]*$modul,$h,"F");
            $x += ($c[$j++]+$c[$j])*$modul;
        }
    }
}
}                                                                                    // FIN DE
CLASSE
?>
```

```
<?php
require('code128.php');

$pdf=new PDF_Code128();
$pdf->AddPage();
$pdf->SetFont('Arial','',10);

//A set
$code='CODE 128';
$pdf->Code128(50,20,$code,80,20);
$pdf->SetXY(50,45);
$pdf->Write(5,'A set: "'.$code.'"');

//B set
$code='Code 128';
$pdf->Code128(50,70,$code,80,20);
$pdf->SetXY(50,95);
$pdf->Write(5,'B set: "'.$code.'"');

//C set
$code='12345678901234567890';
$pdf->Code128(50,120,$code,110,20);
$pdf->SetXY(50,145);
$pdf->Write(5,'C set: "'.$code.'"');

//A,C,B sets
$code='ABCDEFG1234567890AbCdEf';
$pdf->Code128(50,170,$code,125,20);
$pdf->SetXY(50,195);
$pdf->Write(5,'ABC sets combined: "'.$code.'"');

$pdf->Output();
?>
```

EAN-13 Generator

© oliver fpdf.org

```
<?php
require('fpdf.php');

class PDF_EAN13 extends FPDF
{
function EAN13($x, $y, $barcode, $h=16, $w=.35)
{
    $this->Barcode($x,$y,$barcode,$h,$w,13);
}

function UPC_A($x, $y, $barcode, $h=16, $w=.35)
{
```

```
    $this->Barcode($x,$y,$barcode,$h,$w,12);
}

function GetCheckDigit($barcode)
{
    //Compute the check digit
    $sum=0;
    for($i=1;$i<=11;$i+=2)
        $sum+=3*$barcode[$i];
    for($i=0;$i<=10;$i+=2)
        $sum+=$barcode[$i];
    $r=$sum%10;
    if($r>0)
        $r=10-$r;
    return $r;
}

function TestCheckDigit($barcode)
{
    //Test validity of check digit
    $sum=0;
    for($i=1;$i<=11;$i+=2)
        $sum+=3*$barcode[$i];
    for($i=0;$i<=10;$i+=2)
        $sum+=$barcode[$i];
    return ($sum+$barcode[12])%10==0;
}

function Barcode($x, $y, $barcode, $h, $w, $len)
{
    //Padding
    $barcode=str_pad($barcode,$len-1,'0',STR_PAD_LEFT);
    if($len==12)
        $barcode='0'.$barcode;
    //Add or control the check digit
    if(strlen($barcode)==12)
        $barcode.=$this->GetCheckDigit($barcode);
    elseif(!$this->TestCheckDigit($barcode))
        $this->Error('Incorrect check digit');
    //Convert digits to bars
    $codes=array(
        'A'=>array(
            '0'=>'0001101','1'=>'0011001','2'=>'0010011','3'=>'0111101','4'=>'0100011',
            '5'=>'0110001','6'=>'0101111','7'=>'0111011','8'=>'0110111','9'=>'0001011'),
        'B'=>array(
            '0'=>'0100111','1'=>'0110011','2'=>'0011011','3'=>'0100001','4'=>'0011101',
            '5'=>'0111001','6'=>'0000101','7'=>'0010001','8'=>'0001001','9'=>'0010111'),
        'C'=>array(
            '0'=>'1110010','1'=>'1100110','2'=>'1101100','3'=>'1000010','4'=>'1011100',
            '5'=>'1001110','6'=>'1010000','7'=>'1000100','8'=>'1001000','9'=>'1110100')
        );
    $parities=array(
        '0'=>array('A','A','A','A','A','A'),
        '1'=>array('A','A','B','A','B','B'),
        '2'=>array('A','A','B','B','A','B'),
        '3'=>array('A','A','B','B','B','A'),
        '4'=>array('A','B','A','A','B','B'),
        '5'=>array('A','B','B','A','A','B'),
        '6'=>array('A','B','B','B','A','A'),
        '7'=>array('A','B','A','B','A','B'),
        '8'=>array('A','B','A','B','B','A'),
        '9'=>array('A','B','B','A','B','A')
```

```
        );
    $code='101';
    $p=$parities[$barcode[0]];
    for($i=1;$i<=6;$i++)
        $code.=$codes[$p[$i-1]][$barcode[$i]];
    $code.='01010';
    for($i=7;$i<=12;$i++)
        $code.=$codes['C'][$barcode[$i]];
    $code.='101';
    //Draw bars
    for($i=0;$i<strlen($code);$i++)
    {
        if($code[$i]=='1')
            $this->Rect($x+$i*$w,$y,$w,$h,'F');
    }
    //Print text uder barcode
    $this->SetFont('Arial','',12);
    $this->Text($x,$y+$h+11/$this->k,substr($barcode,-$len));
}
}
?>
```

```
<?php
require('ean13.php');
$pdf=new PDF_EAN13();
$pdf->AddPage();
$pdf->EAN13(80,40,'123456789012');
$pdf->Output();
?>
```

Write HTML


```
<?php
//Based on HTML2PDF by Clément Lavoillotte

require('fpdf.php');

//function hex2dec
//returns an associative array (keys: R,G,B) from a hex html code (e.g. #3FE5AA)
function hex2dec($couleur = "#000000"){
    $R = substr($couleur, 1, 2);
    $rouge = hexdec($R);
    $V = substr($couleur, 3, 2);
    $vert = hexdec($V);
    $B = substr($couleur, 5, 2);
    $bleu = hexdec($B);
    $tbl_couleur = array();
    $tbl_couleur['R']=$rouge;
    $tbl_couleur['G']=$vert;
    $tbl_couleur['B']=$bleu;
    return $tbl_couleur;
}

//conversion pixel -> millimeter in 72 dpi
function px2mm($px){
    return $px*25.4/72;
}
```

```
function txtentities($html){
    $trans = get_html_translation_table(HTML_ENTITIES);
    $trans = array_flip($trans);
    return strtr($html, $trans);
}
////////////////////////////////////

class PDF extends FPDF
{
//variables of html parser
protected $B;
protected $I;
protected $U;
protected $HREF;
protected $fontList;
protected $issetfont;
protected $issetcolor;

function __construct($orientation='P', $unit='mm', $format='A4')
{
    //Call parent constructor
    parent::__construct($orientation,$unit,$format);

    //Initialization
    $this->B=0;
    $this->I=0;
    $this->U=0;
    $this->HREF='';

    $this->tableborder=0;
    $this->tdbegin=false;
    $this->tdwidth=0;
    $this->tdheight=0;
    $this->tdalign="L";
    $this->tdbgcolor=false;

    $this->oldx=0;
    $this->oldy=0;

    $this->fontlist=array("arial","times","courier","helvetica","symbol");
    $this->issetfont=false;
    $this->issetcolor=false;
}

////////////////////////////////////
//html parser

function WriteHTML($html)
{
$html=strip_tags($html,"<b><u><i><a><img><p><br><strong><em><font><tr><blockquote><hr><td><tr
><table><sup>"); //remove all unsupported tags
    $html=str_replace("\n",'',$html); //replace carriage returns with spaces
    $html=str_replace("\t",'',$html); //replace carriage returns with spaces
    $a=preg_split('/<(.*)>/U',$html,-1,PREG_SPLIT_DELIM_CAPTURE); //explode the string
    foreach($a as $i=>$e)
    {
        if($i%2==0)
        {
            //Text
            if($this->HREF)
                $this->PutLink($this->HREF,$e);
            elseif($this->tdbegin) {
```

```
                if(trim($e)!='' && $e!=" ") {
$this->Cell($this->tdwidth,$this->tdheight,$e,$this->tableborder,'',$this->tdalign,$this->tdb
gcolor);
                }
                elseif($e==" ") {
$this->Cell($this->tdwidth,$this->tdheight,'',$this->tableborder,'',$this->tdalign,$this->tdb
gcolor);
                }
            }
            else
                $this->Write(5,stripslashes(txtentities($e)));
        }
        else
        {
            //Tag
            if($e[0]=='/')
                $this->CloseTag(strtoupper(substr($e,1)));
            else
            {
                //Extract attributes
                $a2=explode(' ',$e);
                $tag=strtoupper(array_shift($a2));
                $attr=array();
                foreach($a2 as $v)
                {
                    if(preg_match('/([^=]*)=["\']?([^"\']*)/',$v,$a3))
                        $attr[strtoupper($a3[1])]=$a3[2];
                }
                $this->OpenTag($tag,$attr);
            }
        }
    }
}

function OpenTag($tag, $attr)
{
    //Opening tag
    switch($tag){

        case 'SUP':
            if( !empty($attr['SUP']) ) {
                //Set current font to 6pt
                $this->SetFont('','',6);
                //Start 125cm plus width of cell to the right of left margin
                //Superscript "1"
                $this->Cell(2,2,$attr['SUP'],0,0,'L');
            }
            break;

        case 'TABLE': // TABLE-BEGIN
            if( !empty($attr['BORDER']) ) $this->tableborder=$attr['BORDER'];
            else $this->tableborder=0;
            break;
        case 'TR': //TR-BEGIN
            break;
        case 'TD': // TD-BEGIN
            if( !empty($attr['WIDTH']) ) $this->tdwidth=($attr['WIDTH']/4);
            else $this->tdwidth=40; // Set to your own width if you need bigger fixed cells
            if( !empty($attr['HEIGHT']) ) $this->tdheight=($attr['HEIGHT']/6);
            else $this->tdheight=6; // Set to your own height if you need bigger fixed cells
            if( !empty($attr['ALIGN']) ) {
                $align=$attr['ALIGN'];
```

```
                if($align=='LEFT') $this->tdalign='L';
                if($align=='CENTER') $this->tdalign='C';
                if($align=='RIGHT') $this->tdalign='R';
            }
            else $this->tdalign='L'; // Set to your own
            if( !empty($attr['BGCOLOR']) ) {
                $coul=hex2dec($attr['BGCOLOR']);
                    $this->SetFillColor($coul['R'],$coul['G'],$coul['B']);
                    $this->tdbgcolor=true;
                }
            $this->tdbegin=true;
            break;

        case 'HR':
            if( !empty($attr['WIDTH']) )
                $Width = $attr['WIDTH'];
            else
                $Width = $this->w - $this->lMargin-$this->rMargin;
            $x = $this->GetX();
            $y = $this->GetY();
            $this->SetLineWidth(0.2);
            $this->Line($x,$y,$x+$Width,$y);
            $this->SetLineWidth(0.2);
            $this->Ln(1);
            break;
        case 'STRONG':
            $this->SetStyle('B',true);
            break;
        case 'EM':
            $this->SetStyle('I',true);
            break;
        case 'B':
        case 'I':
        case 'U':
            $this->SetStyle($tag,true);
            break;
        case 'A':
            $this->HREF=$attr['HREF'];
            break;
        case 'IMG':
            if(isset($attr['SRC']) && (isset($attr['WIDTH']) || isset($attr['HEIGHT']))) {
                if(!isset($attr['WIDTH']))
                    $attr['WIDTH'] = 0;
                if(!isset($attr['HEIGHT']))
                    $attr['HEIGHT'] = 0;
                $this->Image($attr['SRC'], $this->GetX(), $this->GetY(),
px2mm($attr['WIDTH']), px2mm($attr['HEIGHT']));
            }
            break;
        case 'BLOCKQUOTE':
        case 'BR':
            $this->Ln(5);
            break;
        case 'P':
            $this->Ln(10);
            break;
        case 'FONT':
            if (isset($attr['COLOR']) && $attr['COLOR']!='') {
                $coul=hex2dec($attr['COLOR']);
                $this->SetTextColor($coul['R'],$coul['G'],$coul['B']);
                $this->issetcolor=true;
            }
```

```
            if (isset($attr['FACE']) && in_array(strtolower($attr['FACE']), $this->fontlist))
{
                $this->SetFont(strtolower($attr['FACE']));
                $this->issetfont=true;
            }
            if (isset($attr['FACE']) && in_array(strtolower($attr['FACE']), $this->fontlist)
&& isset($attr['SIZE']) && $attr['SIZE']!='') {
                $this->SetFont(strtolower($attr['FACE']),'',$attr['SIZE']);
                $this->issetfont=true;
            }
            break;
    }
}

function CloseTag($tag)
{
    //Closing tag
    if($tag=='SUP') {
    }

    if($tag=='TD') { // TD-END
        $this->tdbegin=false;
        $this->tdwidth=0;
        $this->tdheight=0;
        $this->tdalign="L";
        $this->tdbgcolor=false;
    }
    if($tag=='TR') { // TR-END
        $this->Ln();
    }
    if($tag=='TABLE') { // TABLE-END
        $this->tableborder=0;
    }

    if($tag=='STRONG')
        $tag='B';
    if($tag=='EM')
        $tag='I';
    if($tag=='B' || $tag=='I' || $tag=='U')
        $this->SetStyle($tag,false);
    if($tag=='A')
        $this->HREF='';
    if($tag=='FONT'){
        if ($this->issetcolor==true) {
            $this->SetTextColor(0);
        }
        if ($this->issetfont) {
            $this->SetFont('arial');
            $this->issetfont=false;
        }
    }
}

function SetStyle($tag, $enable)
{
    //Modify style and select corresponding font
    $this->$tag+=($enable ? 1 : -1);
    $style='';
    foreach(array('B','I','U') as $s) {
        if($this->$s>0)
            $style.=$s;
    }
```

```
    $this->SetFont('',$style);
}

function PutLink($URL, $txt)
{
    //Put a hyperlink
    $this->SetTextColor(0,0,255);
    $this->SetStyle('U',true);
    $this->Write(5,$txt,$URL);
    $this->SetStyle('U',false);
    $this->SetTextColor(0);
}

}//end of class

?>
```

```
<?php
require('html_table.php');

$pdf=new PDF();
$pdf->AddPage();
$pdf->SetFont('Arial','',12);

$html='<table border="1">
<tr>
<td width="200" height="30">cell 1</td><td width="200" height="30" bgcolor="#D0D0FF">cell
2</td>
</tr>
<tr>
<td width="200" height="30">cell 3</td><td width="200" height="30">cell 4</td>
</tr>
</table>';

$pdf->WriteHTML($html);
$pdf->Output();
?>
```