รายการอ้างอิง

1. Adobe Systems Incorporated., *Portable Document Format Reference Manual, version 1.3*. Adobe Systems, 1999, ISBN 0–201–62628–4
2. Adobe Systems Incorporated, *PostScript Language Reference Manual, Third Edition*, Addison-Wesley, 1990, ISBN 0-201-37922-8.
3. Jun-ichi Aoe, *Computer Algorithm: string pattern matching strategies*, IEEE Computer Society Press, 1994, ISBN 0-8186-5461-9 (microfiche)

ภาคผนวก

ภาคผนวก ก

มาตรฐานการเข้ารหัสชุดแบบอักษรไทยในเอกสารพีดีเอฟ

Char	Name	StandardEncoding		MacRomanEncoding		WinAnsiEncoding		PDFDocEncoding	
		Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal
A	A	65	101	65	101	65	101	65	101
Æ	AE	225	341	174	256	198	306	198	306
Á	Aacute	—	—	231	347	193	301	193	301
Â	Acircumflex	—	—	229	345	194	302	194	302
Ä	Adieresis	—	—	128	200	196	304	196	304
À	Agrave	—	—	203	313	192	300	192	300
Å	Aring	—	—	129	201	197	305	197	305
Ã	Atilde	—	—	204	314	195	303	195	303
B	B	66	102	66	102	66	102	66	102
C	C	67	103	67	103	67	103	67	103
Ç	Ccedilla	—	—	130	202	199	307	199	307
D	D	68	104	68	104	68	104	68	104
E	E	69	105	69	105	69	105	69	105
É	Eacute	—	—	131	203	201	311	201	311
Ê	Ecircumflex	—	—	230	346	202	312	202	312
Ë	Edieresis	—	—	232	350	203	313	203	313
È	Egrave	—	—	233	351	200	310	200	310
Đ	Eth	—	—	—	—	208	320	208	320
	Euro	—	—	—	—	128	200	160	240
F	F	70	106	70	106	70	106	70	106
G	G	71	107	71	107	71	107	71	107
H	H	72	110	72	110	72	110	72	110
I	I	73	111	73	111	73	111	73	111
Í	Iacute	—	—	234	352	205	315	205	315
Î	Icircumflex	—	—	235	353	206	316	206	316
Ï	Idieresis	—	—	236	354	207	317	207	317
Ì	Igrave	—	—	237	355	204	314	204	314
J	J	74	112	74	112	74	112	74	112
K	K	75	113	75	113	75	113	75	113
L	L	76	114	76	114	76	114	76	114
	Lslash	232	350	—	—	—	—	149	225
M	M	77	115	77	115	77	115	77	115
N	N	78	116	78	116	78	116	78	116
Ñ	Ntilde	—	—	132	204	209	321	209	321
O	O	79	117	79	117	79	117	79	117
Œ	OE	234	352	206	316	140	214	150	226
Ó	Oacute	—	—	238	356	211	323	211	323
Ô	Ocircumflex	—	—	239	357	212	324	212	324
Ö	Odieresis	—	—	133	205	214	326	214	326
Ò	Ograve	—	—	241	361	210	322	210	322
Ø	Oslash	233	351	175	257	216	330	216	330
Õ	Otilde	—	—	205	315	213	325	213	325
P	P	80	120	80	120	80	120	80	120

PDF 1.3

Char	Name	StandardEncoding		MacRomanEncoding		WinAnsiEncoding		PDFDocEncoding	
		Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal
Q	Q	81	121	81	121	81	121	81	121
R	R	82	122	82	122	82	122	82	122
S	S	83	123	83	123	83	123	83	123
Š	Scaron	—	—	—	—	138	212	151	227
T	T	84	124	84	124	84	124	84	124
Þ	Thorn	—	—	—	—	222	336	222	336
U	U	85	125	85	125	85	125	85	125
Ú	Uacute	—	—	242	362	218	332	218	332
Û	Ucircumflex	—	—	243	363	219	333	219	333
Ü	Udieresis	—	—	134	206	220	334	220	334
Ù	Ugrave	—	—	244	364	217	331	217	331
V	V	86	126	86	126	86	126	86	126
W	W	87	127	87	127	87	127	87	127
X	X	88	130	88	130	88	130	88	130
Y	Y	89	131	89	131	89	131	89	131
Ý	Yacute	—	—	—	—	221	335	221	335
Ÿ	Ydieresis	—	—	217	331	159	237	152	230
Z	Z	90	132	90	132	90	132	90	132
	Zcaron	—	—	—	—	142	216	153	231
a	a	97	141	97	141	97	141	97	141
á	aacute	—	—	135	207	225	341	225	341
â	acircumflex	—	—	137	211	226	342	226	342
´	acute	194	302	171	253	180	264	180	264
ä	adieresis	—	—	138	212	228	344	228	344
æ	ae	241	361	190	276	230	346	230	346
à	agrave	—	—	136	210	224	340	224	340
&	ampersand	38	46	38	46	38	46	38	46
å	aring	—	—	140	214	229	345	229	345
^	asciicircum	94	136	94	136	94	136	94	136
~	asciitilde	126	176	126	176	126	176	126	176
*	asterisk	42	52	42	52	42	52	42	52
@	at	64	100	64	100	64	100	64	100
ã	atilde	—	—	139	213	227	343	227	343
b	b	98	142	98	142	98	142	98	142
\	backslash	92	134	92	134	92	134	92	134
|	bar	124	174	124	174	124	174	124	174
{	braceleft	123	173	123	173	123	173	123	173
}	braceright	125	175	125	175	125	175	125	175
[	bracketleft	91	133	91	133	91	133	91	133
]	bracketright	93	135	93	135	93	135	93	135
	breve	198	306	249	371	—	—	24	30
¦	brokenbar	—	—	—	—	166	246	166	246
•	bullet	183	267	165	245	149	225	128	200
c	c	99	143	99	143	99	143	99	143
ˇ	caron	207	317	255	377	—	—	25	31

PDF 1.3

Char	Name	StandardEncoding		MacRomanEncoding		WinAnsiEncoding		PDFDocEncoding	
		Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal
ç	ccedilla	—	—	141	215	231	347	231	347
¸	cedilla	203	313	252	374	184	270	184	270
¢	cent	162	242	162	242	162	242	162	242
ˆ	circumflex	195	303	246	366	136	210	26	32
:	colon	58	72	58	72	58	72	58	72
,	comma	44	54	44	54	44	54	44	54
©	copyright	—	—	169	251	169	251	169	251
¤	currency	168	250	219	333	164	244	164	244
d	d	100	144	100	144	100	144	100	144
†	dagger	178	262	160	240	134	206	129	201
‡	daggerdbl	179	263	224	340	135	207	130	202
°	degree	—	—	161	241	176	260	176	260
¨	dieresis	200	310	172	254	168	250	168	250
÷	divide	—	—	214	326	247	367	247	367
$	dollar	36	44	36	44	36	44	36	44
˙	dotaccent	199	307	250	372	—	—	27	33
ı	dotlessi	245	365	245	365	—	—	154	232
e	e	101	145	101	145	101	145	101	145
é	eacute	—	—	142	216	233	351	233	351
ê	ecircumflex	—	—	144	220	234	352	234	352
ë	edieresis	—	—	145	221	235	353	235	353
è	egrave	—	—	143	217	232	350	232	350
8	eight	56	70	56	70	56	70	56	70
…	ellipsis	188	274	201	311	133	205	131	203
—	emdash	208	320	209	321	151	227	132	204
–	endash	177	261	208	320	150	226	133	205
=	equal	61	75	61	75	61	75	61	75
ð	eth	—	—	—	—	240	360	240	360
!	exclam	33	41	33	41	33	41	33	41
¡	exclamdown	161	241	193	301	161	241	161	241
f	f	102	146	102	146	102	146	102	146
?	fi	174	256	222	336	—	—	147	223
5	five	53	65	53	65	53	65	53	65
?	fl	175	257	223	337	—	—	148	224
ƒ	florin	166	246	196	304	131	203	134	206
4	four	52	64	52	64	52	64	52	64
/	fraction	164	244	218	332	—	—	135	207
g	g	103	147	103	147	103	147	103	147
ß	germandbls	251	373	167	247	223	337	223	337
`	grave	193	301	96	140	96	140	96	140
>	greater	62	76	62	76	62	76	62	76
«	guillemotleft	171	253	199	307	171	253	171	253
»	guillemotright	187	273	200	310	187	273	187	273
‹	guilsinglleft	172	254	220	334	139	213	136	210
›	guilsinglright	173	255	221	335	155	233	137	211

Char	Name	StandardEncoding		MacRomanEncoding		WinAnsiEncoding		PDFDocEncoding	
		Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal
h	h	104	150	104	150	104	150	104	150
˝	hungarumlaut	205	315	253	375	—	—	28	34
-	hyphen	45	55	45	55	45	55	45	55
ı	i	105	151	105	151	105	151	105	151
í	iacute	—	—	146	222	237	355	237	355
î	icircumflex	—	—	148	224	238	356	238	356
ï	idieresis	—	—	149	225	239	357	239	357
ì	igrave	—	—	147	223	236	354	236	354
j	j	106	152	106	152	106	152	106	152
k	k	107	153	107	153	107	153	107	153
l	l	108	154	108	154	108	154	108	154
<	less	60	74	60	74	60	74	60	74
¬	logicalnot	—	—	194	302	172	254	172	254
	lslash	248	370	—	—	—	—	155	233
m	m	109	155	109	155	109	155	109	155
¯	macron	197	305	248	370	175	257	175	257
	minus	—	—	—	—	—	—	138	212
µ	mu	—	—	181	265	181	265	181	265
×	multiply	—	—	—	—	215	327	215	327
n	n	110	156	110	156	110	156	110	156
9	nine	57	71	57	71	57	71	57	71
ñ	ntilde	—	—	150	226	241	361	241	361
#	numbersign	35	43	35	43	35	43	35	43
o	o	111	157	111	157	111	157	111	157
ó	oacute	—	—	151	227	243	363	243	363
ô	ocircumflex	—	—	153	231	244	364	244	364
ö	odieresis	—	—	154	232	246	366	246	366
œ	oe	250	372	207	317	156	234	156	234
˛	ogonek	206	316	254	376	—	—	29	35
ò	ograve	—	—	152	230	242	362	242	362
1	one	49	61	49	61	49	61	49	61
½	onehalf	—	—	—	—	189	275	189	275
¼	onequarter	—	—	—	—	188	274	188	274
¹	onesuperior	—	—	—	—	185	271	185	271
ª	ordfeminine	227	343	187	273	170	252	170	252
º	ordmasculine	235	353	188	274	186	272	186	272
ø	oslash	249	371	191	277	248	370	248	370
õ	otilde	—	—	155	233	245	365	245	365
p	p	112	160	112	160	112	160	112	160
¶	paragraph	182	266	166	246	182	266	182	266
(	parenleft	40	50	40	50	40	50	40	50
)	parenright	41	51	41	51	41	51	41	51
%	percent	37	45	37	45	37	45	37	45
.	period	46	56	46	56	46	56	46	56
·	periodcentered	180	264	225	341	183	267	183	267

Char	Name	StandardEncoding		MacRomanEncoding		WinAnsiEncoding		PDFDocEncoding	
		Decimal	Octal	Decimal	Octal	Decimal	Octal	Decimal	Octal
‰	perthousand	189	275	228	344	137	211	139	213
+	plus	43	53	43	53	43	53	43	53
±	plusminus	—	—	177	261	177	261	177	261
q	q	113	161	113	161	113	161	113	161
?	question	63	77	63	77	63	77	63	77
¿	questiondown	191	277	192	300	191	277	191	277
"	quotedbl	34	42	34	42	34	42	34	42
„	quotedblbase	185	271	227	343	132	204	140	214
“	quotedblleft	170	252	210	322	147	223	141	215
”	quotedblright	186	272	211	323	148	224	142	216
`	quoteleft	96	140	212	324	145	221	143	217
'	quoteright	39	47	213	325	146	222	144	220
‚	quotesinglbase	184	270	226	342	130	202	145	221
'	quotesingle	169	251	39	47	39	47	39	47
r	r	114	162	114	162	114	162	114	162
®	registered	—	—	168	250	174	256	174	256
°	ring	202	312	251	373	176	260	30	36
s	s	115	163	115	163	115	163	115	163
š	scaron	—	—	—	—	154	232	157	235
§	section	167	247	164	244	167	247	167	247
;	semicolon	59	73	59	73	59	73	59	73
7	seven	55	67	55	67	55	67	55	67
6	six	54	66	54	66	54	66	54	66
/	slash	47	57	47	57	47	57	47	57
	space	32	40	32, 202	40,312	32	40	32	40
£	sterling	163	243	163	243	163	243	163	243
t	t	116	164	116	164	116	164	116	164
þ	thorn	—	—	—	—	254	376	254	376
3	three	51	63	51	63	51	63	51	63
¾	threequarters	—	—	—	—	190	276	190	276
³	threesuperior	—	—	—	—	179	263	179	263
~	tilde	196	304	247	367	152	230	31	37
™	trademark	—	—	170	252	153	231	146	222
2	two	50	62	50	62	50	62	50	62
²	twosuperior	—	—	—	—	178	262	178	262
u	u	117	165	117	165	117	165	117	165
ú	uacute	—	—	156	234	250	372	250	372
û	ucircumflex	—	—	158	236	251	373	251	373
ü	udieresis	—	—	159	237	252	374	252	374
ù	ugrave	—	—	157	235	249	371	249	371
	underscore	95	137	95	137	95	137	95	137
v	v	118	166	118	166	118	166	118	166
w	w	119	167	119	167	119	167	119	167
x	x	120	170	120	170	120	170	120	170
y	y	121	171	121	171	121	171	121	171

ภาคผนวก ข

โปรแกรมส่วนจำเพาะการค้นข้อความไทยในเอกสารพีดีเอฟ

```
//////////////////////////////////////////////////////////////////////
// PDFPlugins.cpp: implementation of the CPDFPlugins class.
//
//////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "ThaiFind.h"
#include "PDFPlugins.h"
#include "ThaiPDF.h"
#include "FindDlg.h"
#include "AboutDlg.h"
#include "ProgressDlg.h"

#if WIN_PLATFORM
#include <windows.h>
#include "resource.h"
#endif

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

static void *GetToolIcon(int ThFindBmp)
{
	HBITMAP icon;
	HINSTANCE gHINSTANCE;

	gHINSTANCE = AfxGetResourceHandle( );
	switch (ThFindBmp) {
	case 1:
		icon = LoadBitmap(gHINSTANCE, MAKEINTRESOURCE(IDB_THFINDBACK));
		break;
	case 2:
		icon = LoadBitmap(gHINSTANCE, MAKEINTRESOURCE(IDB_THFIND));
		break;
	case 3:
		icon = LoadBitmap(gHINSTANCE, MAKEINTRESOURCE(IDB_THFINDFOR));
		break;
	}
	return (void *)icon;
}

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CPDFPlugins::CPDFPlugins()
{

}

CPDFPlugins::~CPDFPlugins()
{

}
```

```
ACCB1 ASBool ACCB2 PIHandshake(ASUns32 handshakeVersion, void *handshakeData)
{
        if (handshakeVersion == HANDSHAKE_V0200) {
                /* Cast handshakeData to the appropriate type */
                PIHandshakeData_V0200 *hsData = (PIHandshakeData_V0200 *)handshakeData;

                /* Set the name we want to go by */
                hsData->extensionName = ASAtomFromString("ENUM_WORDS_THAIFIND_PGIN");

                /* If you replace host functionality, do so in here */
                hsData->importReplaceAndRegisterCallback = NULL;

                /* Set the initialization routine */
                hsData->initCallback = ASCallbackCreateProto(PIInitProcType,
                                        CPDFPlugins::InitMenuItem);

                /* No unload proc */
                hsData->unloadCallback = ASCallbackCreateProto(PIUnloadProcType,
                                                CPDFPlugins::UnloadMenuItem);

                /* All done */
                return true;

        } /* Each time the handshake version changes, add a new "else if" branch */

        /* If we reach here, then we were passed a handshake version number we
         * don't know about.  This shouldn't ever happen since our main() routine
         * chose the version number.
         */
        return(false);
}

ACCB1 ASBool ACCB2 CPDFPlugins::InitMenuItem()
{
        AVMenubar MenuBar;
        AVMenu ThSrchExtsMenu;
        AVMenu AboutThSrchExtsMenu;
        AVMenuItem AboutThaiSrchMenuItem;
        AVMenuItem ThaiSrchMenuItem;
        AVMenuItem ThaiSrchForWardMenuItem;
        AVMenuItem ThaiSrchBackWardMenuItem;
        AVComputeEnabledProc EnableProc;

        AVToolBar ToolBar =AVAppGetToolBar();
        AVToolButton ThFindButton = NULL;
        AVToolButton FindButton = NULL;

        /* Acquire pull-down menu handles. */
        MenuBar = AVAppGetMenubar();
        if (!MenuBar) return(false);
        ThSrchExtsMenu = AVMenubarAcquireMenuByName(MenuBar, STR_MENUBAR_GROUP);
        if (!ThSrchExtsMenu) return(false);
        AboutThSrchExtsMenu     = AVMenubarAcquireMenuByName(MenuBar,
                                        STR_ABOUT_PLUGINS);
        if (!AboutThSrchExtsMenu) return(false);

        /* Install command in Extensions pull-down menu */
        ThaiSrchMenuItem = AVMenuItemNew(STR_MENUITEM_TITLE,
                                STR_MENUITEM_NAME,
                                NULL,
                                false,
                                'F',
                                11,                             /* Flags */
                                NULL,
                                gExtensionID);
```

```
ThaiSrchForWardMenuItem = AVMenuItemNew("ค้น ต่อไป",
                        "SPCpin:THSRCHFW",
                        NULL,
                        false,
                        'G',
                        11,                             /* Flags */
                        NULL,
                        gExtensionID);
ThaiSrchBackWardMenuItem = AVMenuItemNew("ค้น ย้อนหลัง",
                        "SPCpin:THSRCHBW",
                        NULL,
                        false,
                        'B',
                        11,                             /* Flags */
                        NULL,
                        gExtensionID);
if (!ThaiSrchMenuItem)
        return(false);
if(!ThaiSrchForWardMenuItem)
        AVAlertNote("ForWardMenu Fail!");

AVMenuItemSetExecuteProc(ThaiSrchMenuItem,
                        ASCallbackCreateProto(AVExecuteProc,
                                CPDFPlugins::ThaiTextFind),
                        NULL);
AVMenuItemSetExecuteProc(ThaiSrchForWardMenuItem,
                        ASCallbackCreateProto(AVExecuteProc,
                                CPDFPlugins::ThaiTextFindForward),
                        NULL);
AVMenuItemSetExecuteProc(ThaiSrchBackWardMenuItem,
                        ASCallbackCreateProto(AVExecuteProc,
                                CPDFPlugins::ThaiTextFindBackward),
                        NULL);

EnableProc = ASCallbackCreateProto(AVComputeEnabledProc,
                CPDFPlugins::EnableMyMenuItem);
AVMenuItemSetComputeEnabledProc(ThaiSrchMenuItem, EnableProc, NULL);
AVMenuItemSetComputeEnabledProc(ThaiSrchForWardMenuItem,
        ASCallbackCreateProto(AVComputeEnabledProc,
                CPDFPlugins::EnableForWardMenuItem),
        NULL);
AVMenuItemSetComputeEnabledProc(ThaiSrchBackWardMenuItem,
        ASCallbackCreateProto(AVComputeEnabledProc,
                CPDFPlugins::EnableForWardMenuItem),
        NULL);

ThFindButton = AVToolButtonNew(ASAtomFromString(STR_TOOLBUTTON_NAME),
                GetToolIcon(2), false, false);
AVToolButtonSetHelpText(ThFindButton,"ThaiFind2803 (Ctrl+Shift+F)");
AVToolButtonSetExecuteProc(ThFindButton,
        ASCallbackCreateProto(AVExecuteProc, CPDFPlugins::ThaiTextFind), NULL);
AVToolButtonSetComputeEnabledProc(ThFindButton, EnableProc, NULL);
FindButton = AVToolBarGetButtonByName(ToolBar, ASAtomFromString
                (STR_AFTER_TOOLBAR));
AVToolBarAddButton(ToolBar, ThFindButton, FALSE, FindButton);

AVMenuItem FindMenuItem = AVMenubarAcquireMenuItemByName(MenuBar,
                                "endSelectGroup");
int FindMenuIndex = AVMenuGetMenuItemIndex(ThSrchExtsMenu,FindMenuItem);

/* Insert menu item in pull-down. */
AVMenuAddMenuItem(ThSrchExtsMenu, ThaiSrchMenuItem, FindMenuIndex+1);
AVMenuAddMenuItem(ThSrchExtsMenu, ThaiSrchForWardMenuItem, FindMenuIndex+2);
AVMenuAddMenuItem(ThSrchExtsMenu, ThaiSrchBackWardMenuItem, FindMenuIndex+3);
AVMenuRelease(ThSrchExtsMenu);
```

```
	/* Install About command in Help pull-down menu. */
	AboutThaiSrchMenuItem = AVMenuItemNew(STR_MENUITEM_ABOUT_TITLE,
				STR_MENUITEM_ABOUT_NAME,
				NULL,
				false,
				NO_SHORTCUT,
				0,                                  /* Flags  */
				NULL,
				gExtensionID);
	if (!AboutThaiSrchMenuItem) return(false);
	AVMenuItemSetExecuteProc(AboutThaiSrchMenuItem,
			ASCallbackCreateProto(AVExecuteProc, CPDFPlugins::AboutThaiFind),
			NULL);

	/* Insert menu item. */
	AVMenuAddMenuItem(AboutThSrchExtsMenu, AboutThaiSrchMenuItem,
		APPEND_MENUITEM);
	AVMenuRelease(AboutThSrchExtsMenu);

	return true;
}

ACCB1 ASBool ACCB2 CPDFPlugins::UnloadMenuItem()
{
	return true;
}

ASFile FileName;
CString lastPattern;
ACCB1 ASBool ACCB2 CPDFPlugins::EnableMyMenuItem(void *data)
{
	if( AVAppGetActiveDoc() ){
		AVDoc CurrentAVDoc = AVAppGetActiveDoc();
		PDDoc CurrentPDDoc = AVDocGetPDDoc(CurrentAVDoc);
		ASFile DocFile = PDDocGetFile(CurrentPDDoc);
		if(DocFile!=FileName){
			FileName=DocFile;
			AVDocClearSelection(CurrentAVDoc, true);

		}
		return true;
	}
	else {
		lastPattern = "";
		return false;
	}
}

ACCB1 ASBool ACCB2 CPDFPlugins::EnableForWardMenuItem(void *data)
{
	if( AVAppGetActiveDoc() ){
		if(!lastPattern.IsEmpty())
			return true;
		else
			return false;
	}
	else
		return false;
}

ACCB1 void ACCB2 CPDFPlugins::AboutThaiFind(void *data)
{
	CAboutDlg AboutDlg;
	int Ret = AboutDlg.DoModal();
}
```

```
ACCB1 void ACCB2 CPDFPlugins::ThaiTextFind(void *data)
{
	CFindDlg ThFindDlg;
	CThaiPDF ThaiPDF;
	int Ret=0;

	ThFindDlg m_strPattern = lastPattern;
	Ret = ThFindDlg.DoModal();
	while(ThFindDlg.m_strPattern=="") {
		if(Ret==2)
			ThFindDlg.m_strPattern = "~?><";
		if(ThFindDlg.m_strPattern=="") {
			AVAlertNote("กรุณาใส่ข้อความที่ต้องการค้น");
			Ret = ThFindDlg.DoModal();
		}
	}
	if(Ret == 1)
	{
		lastPattern = ThFindDlg.m_strPattern;
		if(ThFindDlg.m_isBackWard)
			ThaiPDF.FindBackward(lastPattern);
		else
			ThaiPDF.FindForward(lastPattern);
	}
}

ACCB1 void ACCB2 CPDFPlugins::ThaiTextFindForward(void *data)
{
	if(lastPattern=="")
			AVAlertNote("ท่านยังไม่ได้ให้ข้อความที่ต้องการค้น\n
				กรุณากดปุ่มThaiFindแล้วพิมพ์ข้อความที่ต้องการค้น");
	else {
		CThaiPDF ThaiPDF;
		ThaiPDF.FindForward(lastPattern);
	}
}

ACCB1 void ACCB2 CPDFPlugins::ThaiTextFindBackward(void *data)
{
	if(lastPattern=="")
			AVAlertNote("ท่านยังไม่ได้ให้ข้อความที่ต้องการค้น\n
				กรุณากดปุ่มThaiFindแล้วพิมพ์ข้อความที่ต้องการค้น");
	else {
		CThaiPDF ThaiPDF;
		ThaiPDF.FindBackward(lastPattern);
	}
}
```

```
////////////////////////////////////////////////////////////
// AboutDlg.cpp : implementation of the CAboutDlg Class
//
////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "ThaiFind.h"
#include "AboutDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog


CAboutDlg::CAboutDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CAboutDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CAboutDlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}


void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)

	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg message handlers

void CAboutDlg::OnOK()
{

	CDialog::OnOK();
}

BOOL CAboutDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}
```

```
////////////////////////////////////////////////////////////
// FindDlg.cpp  implementation of the CFindDlg class
//
////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "ThaiFind.h"
#include "ThaiPDF.h"
#include "FindDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////////////////////////////////////////////
// CFindDlg dialog


CFindDlg::CFindDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CFindDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CFindDlg)
        m_strPattern = _T("");
        m_isBackWard = FALSE;
        //}}AFX_DATA_INIT
}


void CFindDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CFindDlg)
        DDX_Text(pDX, IDC_PATTERN, m_strPattern);
        DDX_Check(pDX, IDC_CHECK_BACKWARD, m_isBackWard);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CFindDlg, CDialog)
        //{{AFX_MSG_MAP(CFindDlg)
        ON_BN_CLICKED(IDC_CHECK_BACKWARD, OnCheckBackward)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


void CFindDlg::OnCheckBackward()
{
        m_isBackWard  = !m_isBackWard;

}
```

```
//////////////////////////////////////////////////////////////////////
//
// ChaiPDF.cpp: implementation of the CThaiPDF class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "ThaiPDF.h"
#include "ThaiFind.h"
#include "FindDlg.h"
#include "ProgressDlg.h"
#include "FontRegDlg.h"
#include "PDExpT.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

//**** Thai Alphabet table for Text Extraction ****

unsigned char tblPlus35[256] = {0,10,13,32,'!','"','#','$','%','&','\',  \
        '(',')','*','+',',','-','.','/','0','1','2','3','4','5',    \
        '6','7','8','9',':',';','<','=','>','?','@','A','B','C',    \
        'D','E','F','G','H','I','J','K','L','M','N','O','P','Q',    \
        'R','S','T','U','V','W','X','Y','Z','[','\\',']','^','_',   \
        '`','a','b','c','d','e','f','g','h','i','j','k','l','m',    \
        'n','o','p','q','r','s','t','u','v','w','x','y','z','{',    \
        '|','}','~',127,175,212,213,214,215,133,232,233,234,235,    \
        236,232,233,234,235,236,173,145,146,147,148,149,150,151,     \
        209,153,231,232,233,234,235,236,160,161,162,163,164,165,     \
        166,167,168,169,170,171,172,173,174,175,176,177,178,179,     \
        180,181,182,183,184,185,186,187,188,189,190,191,192,193,     \
        194,195,196,197,198,199,200,201,202,203,204,205,206,207,     \
        208,209,210,211,212,213,214,215,216,217,218,219,220,223,     \
        224,225,226,227,228,229,230,231,232,233,234,235,236,237,     \
        238,239,240,241,242,243,244,245,246,247,248,249,250,251,     \
        216,217,254,'x','x','x','x','x','x','x','x','x','x','x',    \
        'x','x','x','x','x','x','x','x','x','x','x','x','x'};

unsigned char tblThaiCh[256] = { \
          0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,                  \
          16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,                       \
          ' ','!','"','#','$','%','&','\'','(',')','*','+',',','-','.','/',     \
          '0','1','2','3','4','5','6','7','8','9',':',';','<','=','>','?',     \
          '@','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O',     \
          'P','Q','R','S','T','U','V','W','X','Y','Z','[','\\',']','^','_',    \
          '`','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o',     \
          'p','q','r','s','t','u','v','w','x','y','z','{','|','}','~',127,     \
          176,212,213,214,215,133,232,233,234,235,236,232,233,234,235,236,      \
          173,145,146,147,148,149,150,151,209,153,231,232,233,234,235,236,      \
          160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,      \
          176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,      \
          192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,      \
          208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,      \
          224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,      \
          240,241,242,243,244,245,246,247,248,249,250,251,216,217,254,255};
```

```
unsigned char tblF7[30] = {\
        176,212,213,214,215,232,233,234,235,236,232,233,234,235,236,173,  \
        209,254,231,232,233,234,235,236,216,217,254,219};

//*************************************************

//**** Global variable of Thai Text Extraction ****
CProgressDlg PgressDlg;
CString gblThaiTxt;
bool CancelByButton=false;

int m_fontEnc=2;

CString strfontName;
CString strBuffType2;
CString strBuffType3;
CString fontList[100];
//*************************************************

CThaiPDF::CThaiPDF()
{
        CurrentAVDoc = AVAppGetActiveDoc();
        CurrentPDDoc = AVDocGetPDDoc(CurrentAVDoc);
        WordFinder = PDDocCreateWordFinder(CurrentPDDoc,NULL,NULL,NULL,
                                                0,WXE_PDF_ORDER,NULL);
        WordFinderUCS = PDDocCreateWordFinderUCS(CurrentPDDoc, 0,
                                                WXE_PDF_ORDER, NULL);
        FontName = strfontName;
        strType2 = strBuffType2;
        strType3 = strBuffType3;
        fontList[0] = "NULL";
}

CThaiPDF::~CThaiPDF()
{
        PDWordFinderDestroy(WordFinder);
        PDWordFinderDestroy(WordFinderUCS);
}

void CThaiPDF::HiliteText(PDDoc CurrentPDDoc, int CurrentPageNum, PDWord word, int NumWord)
{
        AVDoc CurrentAVDoc = AVAppGetActiveDoc();
        PDPage CurrentPDPage;
        PDTextSelect TextSelection;
        HiliteEntry Hilite;

        CurrentPDPage = PDDocAcquirePage(CurrentPDDoc, CurrentPageNum);
        Hilite.offset = PDWordGetCharOffset(word);
        Hilite.length = NumWord;

        TextSelection = PDTextSelectCreatePageHilite(CurrentPDPage, &Hilite, 1);
        AVDocSetSelection(CurrentAVDoc, ASAtomFromString("Text"), (void *)TextSelection, true);
        AVDocShowSelection(CurrentAVDoc);
        PDPageRelease(CurrentPDPage);

}

void CThaiPDF::FindForward(CString pattern)
{
        CancelByButton=false;
        CFindDlg ThFindDlg;
        CString strWord,strTemp=pattern;

        PDWord word, wInfo, wInfoUCS;
        ASInt32 Count, CountUCS;
        PDFont tmpfont;
```

```
int startPOS=-1;
int idxWord;
int lclPosFound = -1;
int NumWord=0, NumHilite=0;
BOOL SubPattern=false;
int i=1, range=0, percent=0;

PDPage CurrentPDPage = AVPageViewGetPage(AVDocGetPageView(CurrentAVDoc));
int CurrentPageNum = PDPageGetNumber(CurrentPDPage);
int TotalPage = PDDocGetNumPages(CurrentPDDoc);

//***** Get Position Word Hilite **********
PDTextSelect ts;
if( ASAtomFromString("Text") == AVDocGetSelectionType(CurrentAVDoc) ) {
        ts = (PDTextSelect)AVDocGetSelection(CurrentAVDoc);
        if(ts != NULL) {
                PDTextSelectRangeRec range;
                PDTextSelectGetRange(ts, 0, &range);
                idxWord = range.end;
        }
        else
                idxWord=0;
}
else
        idxWord=0;
//****************************************

range = TotalPage - CurrentPageNum;
PgressDlg.Create(IDD_PROGRESS, NULL);
PgressDlg.SetDlgItemText(IDC_MSGPATTERN,(LPCTSTR)strTemp);
PgressDlg.SetDlgItemInt(IDC_CURRENT_PAGE, CurrentPageNum + 1, true);

while( CurrentPageNum < TotalPage && lclPosFound == -1 && !CancelByButton) {
        PDWordFinderAcquireWordList(WordFinder, CurrentPageNum, &wInfo,
                                        NULL, NULL, &Count);
        PDWordFinderAcquireWordList(WordFinderUCS, CurrentPageNum, &wInfoUCS,
                                        NULL, NULL, &CountUCS);
        // **** while finding text should getfocus on Cancel button ****
        percent = (i++)*100/range;
        PgressDlg.SetDlgItemInt(IDC_CURRENT_PAGE, CurrentPageNum + 1, true);
        PgressDlg.m_Progress.SetPos(percent);
        PgressDlg.SetDlgItemInt(IDC_PERCENT, percent, true);
        PgressDlg.PeekAndPump();
        if(PgressDlg.Cancelled())
                break;

        while( idxWord < Count && lclPosFound == -1 && !CancelByButton){
                //*** Reg Font Encoding *****
                word = PDWordFinderGetNthWord(WordFinder, idxWord);
                PDStyle style = PDWordGetNthCharStyle(WordFinder,word, 0);
                PDFont font = PDStyleGetFont(style);
                /***** Font Encoding Regconize *******/
                if(tmpfont!=font) {
                        bool canFind4=false;
                        tmpfont = font;
                        m_fontEnc = RegFont(WordFinder, word, font);
                        if(m_fontEnc==5)
                                Count = CountUCS;
                        if(m_fontEnc==4)
                                canFind4 = canFindType4(font, pattern);
                        if(!hasAVAlert(font)
                                &&(m_fontEnc==1 || !canFind4 || m_fontEnc==5)) {
                        if(userConfirm(m_fontEnc, font))
                                CancelByButton = false;
                        else
                                CancelByButton = true;
                        }
```

```
                }
                //**************************
                if(m_fontEnc==5)
                        word = PDWordFinderGetNthWord(WordFinderUCS, idxWord);

                strWord = ThaiTextExtract(word, m_fontEnc);
                strWord.Remove(' ');
                //******* Find Text from PDWord(s)  **********
                if(pattern.GetAt(0)>0 && pattern.GetAt(1)>0) {
                        //*** Dont' diffent between lower and UPPER ***
                        pattern.MakeUpper();
                        strWord.MakeUpper();
                        lclPosFound = strWord.Find(pattern);
                        if(lclPosFound > -1) {
                                NumWord++;
                                NumHilite = NumHilite + PDWordGetLength(word);
                        }
                }
                else {
                        lclPosFound = ThaiPatternMatching(pattern, strWord, SubPattern);
                        lclPosFound = pattern.GetLength();
                        if(lclPosFound < pattern.GetLength()){
                                if(lclPosFound > 0){
                                        SubPattern=true;
                                        NumWord++;
                                        NumHilite += PDWordGetLength(word);
                                        pattern = pattern.Right(pattern.GetLength()-
                                                                lclPosFound);
                                        lclPosFound=-1;
                                }
                                else {
                                        if(SubPattern) {
                                                if(strWord == "")
                                                        NumWord++;
                                                else {
                                                        SubPattern=false;
                                                        pattern=strTemp;
                                                        NumWord=0;
                                                        NumHilite=0;
                                                }
                                        }
                                        else {
                                                NumWord=0;
                                                NumHilite=0;
                                        }
                                        lclPosFound = -1;
                                }
                        }
                        else {
                                NumHilite = NumHilite + PDWordGetLength(word);
                                word = PDWordFinderGetNthWord(WordFinder,
                                                        idxWord-NumWord);
                        }
                }
                //*****************  END  **********************
                idxWord++;
        }

        PDWordFinderReleaseWordList(WordFinder, CurrentPageNum);
        if( lclPosFound == -1 ) {
                CurrentPageNum++;
                idxWord=0;
        }

        if( CurrentPageNum >= TotalPage && !CancelByButton ) {
                PgressDlg.DestroyWindow();
```

```
                        if(AVAlertConfirm("สิ้นสุดเอกสารแล้ว ท่านต้องการย้อนกลับไปค้นใหม่หรือไม่ ?")) {
                                PgressDlg.Create(IDD_PROGRESS, NULL);
                                PgressDlg.SetDlgItemText(IDC_MSGPATTERN,
                                                        (LPCTSTR)strTemp);
                                CurrentPageNum = 0;
                                range = TotalPage - CurrentPageNum;
                                i=0;
                                idxWord = 0;
                        }
                }
        }

        if(lclPosFound != -1){
                idxWord-=NumWord;
                gblThaiTxt.Empty();
                SubPattern=false;
                if(m_fontEnc == 5)
                        NumHilite = NumHilite / 2;
                PgressDlg.DestroyWindow();
                HiliteText(CurrentPDDoc, CurrentPageNum, word, NumHilite);
        }

        NumWord=0;
        NumHilite=0;

}

void CThaiPDF::FindBackward(CString pattern)
{
        CancelByButton=false;
        CFindDlg ThFindDlg;
        CString strWord,strTemp=pattern;

        PDWord word, wInfo, wInfoUCS;
        ASInt32 Count, CountUCS;
        PDFont tmpfont;

        int idxWord;
        int lclPosFound = -1;
        int NumWord=0, NumHilite=0;
        BOOL SubPattern=false;
        int i=1, range=0, percent=0;

        PDPage CurrentPDPage = AVPageViewGetPage(AVDocGetPageView(CurrentAVDoc));
        int CurrentPageNum = PDPageGetNumber(CurrentPDPage);
        int TotalPage = PDDocGetNumPages(CurrentPDDoc);

        //***** Get Position Word Hilite *************
        PDTextSelect ts;
        if( ASAtomFromString("Text") == AVDocGetSelectionType(CurrentAVDoc) ) {
                ts = (PDTextSelect)AVDocGetSelection(CurrentAVDoc);
                if(ts != NULL) {
                        PDTextSelectRangeRec range;
                        PDTextSelectGetRange(ts, 0, &range);
                        idxWord = range.start-1;
                }
                else
                        idxWord=0;
        }
        else
                idxWord=0;
        //*****************************************

        range = CurrentPageNum + 1;
        PgressDlg.Create(IDD_PROGRESS, NULL);
        PgressDlg.SetDlgItemText(IDC_MSGPATTERN,(LPCTSTR)strTemp);
```

```
while( CurrentPageNum > -1 && IclPosFound == -1 && !CancelByButton) {
        PDWordFinderAcquireWordList(WordFinder, CurrentPageNum, &wInfo,
                                        NULL, NULL, &Count),
        PDWordFinderAcquireWordList(WordFinderUCS, CurrentPageNum, &wInfoUCS,
                                        NULL, NULL, &CountUCS);
        // **** while finding text should getfocus on Cancel button ****
        percent = (i++)*100/range;
        PgressDlg.SetDlgItemInt(IDC_CURRENT_PAGE, CurrentPageNum + 1, true);
        PgressDlg.m_Progress SetPos(percent),
        PgressDlg SetDlgItemInt(IDC_PERCENT, percent, true);
        PgressDlg.PeekAndPump();
        if(PgressDlg.Cancelled())
                break;

        while( idxWord > -1 && IclPosFound == -1 && !CancelByButton){

                //*** Reg Font Encoding *****
                word = PDWordFinderGetNthWord(WordFinder, idxWord);
                PDStyle style = PDWordGetNthCharStyle(WordFinder,word, 0);
                PDFont font = PDStyleGetFont(style);
                /***** Font Encoding Regconize *******/
                if(tmpfont!=font) {
                        bool canFind4=false;
                        tmpfont = font;
                        m_fontEnc = RegFont(WordFinder, word, font);
                        if(m_fontEnc==5)
                                Count = CountUCS;
                        if(m_fontEnc==4)
                                canFind4 = canFindType4(font, pattern);
                        if(!hasAVAlert(font)
                                &&(m_fontEnc==1 || !canFind4 || m_fontEnc==5)) {
                                if(userConfirm(m_fontEnc, font))
                                        CancelByButton = false;
                                else
                                        CancelByButton = true;
                        }
                }
                //*************************
                if(m_fontEnc==5)
                        word = PDWordFinderGetNthWord(WordFinderUCS, idxWord);
                //******* Text Extraction and Translate to Thai Text **********
                strWord = ThaiTextExtract(word, m_fontEnc);
                strWord.Remove(' ');
                //*********************************************************

                //******* Find Text from PDWord(i)  **********
                if(pattern.GetAt(0)>0 && pattern.GetAt(1)>0) {
                        //*** Ignor between lower and UPPER ***
                        pattern.MakeUpper();
                        strWord.MakeUpper();

                        IclPosFound = strWord.Find(pattern);
                        if(IclPosFound > -1) {
                                NumWord++;
                                NumHilite = NumHilite + PDWordGetLength(word);
                        }
                }
                else {
                        IclPosFound = ThaiPatternMatching(pattern, strWord, SubPattern);
                        IclPosFound = pattern.GetLength();


                        if(IclPosFound < pattern.GetLength()){
                                if(IclPosFound > 0){
                                        SubPattern=true;
                                        NumWord++;
```

```
                                        NumHilite += PDWordGetLength(word);
                                        pattern = pattern.Right(pattern.GetLength()-
                                                                        lclPosFound);
                                        lclPosFound=-1;
                                }
                                else {
                                        if(SubPattern) {
                                                if(strWord == "")
                                                        NumWord++;
                                                else {
                                                        SubPattern=false;
                                                        pattern=strTemp;
                                                }
                                        }
                                        lclPosFound = -1;
                                }
                        }
                        else {
                                word = PDWordFinderGetNthWord(WordFinder,
                                                                        idxWord-NumWord);
                                NumWord++;
                                NumHilite = NumHilite + PDWordGetLength(word);
                                if(SubPattern)
                                        SubPattern=false;
                        }
                }

                //*****************  END  **********************

                if(SubPattern)
                        idxWord++;
                else {
                        idxWord = idxWord - NumWord;
                        if(lclPosFound == -1){
                                NumWord=0;
                                NumHilite=0;
                                idxWord--;
                        }
                }
        }

        PDWordFinderReleaseWordList(WordFinder, CurrentPageNum);

        if( lclPosFound == -1 ) {
                CurrentPageNum--;
                if( CurrentPageNum < 0 ) {
                        PgressDlg.DestroyWindow();
                        if(AVAlertConfirm("สิ้นสุดเอกสารแล้ว ท่านต้องการย้อนกลับไปค้นใหม่หรือไม่?")) {
                                PgressDlg.Create(IDD_PROGRESS, NULL);
                                PgressDlg.SetDlgItemText(IDC_MSGPATTERN,
                                                                (LPCTSTR)strTemp);
                                CurrentPageNum = TotalPage-1;
                                range = CurrentPageNum + 1;
                                i=1;
                        }
                }
                if( CurrentPageNum > -1 ){
                        PDWordFinderAcquireWordList(WordFinder, CurrentPageNum,
                                                        &wInfo, NULL, NULL, &Count);
                        idxWord=Count-1;
                }
        }
}
```

```
		if(IclPosFound != -1){
			idxWord++;
			gblThaiTxt.Empty();
			SubPattern=false;
			if(m_fontEnc == 5)
				NumHilite = NumHilite / 2;
			PgressDlg.DestroyWindow();
			HiliteText(CurrentPDDoc, CurrentPageNum, word, NumHilite);
		}
		NumWord=0;
		NumHilite=0;

}

bool ISsaraAum=false;
int CThaiPDF::ThaiPatternMatching(CString Pattern, CString Text, BOOL SubPattern)
{
		int i,j=0;
		int NumChar=0;
		unsigned char NowChText, NowChPattern;

		i=0;
		while(i<Text.GetLength()) {
			NowChText = Text.GetAt(i);
			NowChPattern = Pattern.GetAt(j);

			if(SubPattern && Text.GetLength()==0)		// Case Text equal "\0", length is zero
				return 0;

			if(ISsaraAum) {
				if(NowChText == 0xD2) {
					if(NowChPattern==0xD3){
						i++;
						NowChText=211;
						ISsaraAum=false;
					}
					else {
						i++;
						if(i<Text.GetLength())
							NowChText = Text.GetAt(i);
						ISsaraAum=false;
					}
				}
			}
			if(NowChText==0xED){
				if(NowChPattern==0xD3){
					i++;
					NowChText=211;
					ISsaraAum=true;
				}
				else
					ISsaraAum=false;
			}
			if(  NowChText == NowChPattern )
				j++;
			else {
				if(j>0 && !ISsaraAum){
					if(NowChText==(unsigned char)Pattern.GetAt(0))
						j=1;
					else{
						if(NowChText != 32)
							j=0;
					}
				}
				if(SubPattern)
					return j;
```

```
                }
                if(j==Pattern.GetLength())
                        return j;
                i++;
        }

        return j;
}

int CThaiPDF::RegFontEncode(PDFont font, char buff[])
{
        char fontName[200]="";
        PDFontGetName(font, fontName, 200);
        CString strFontName=(LPCTSTR)fontName;
        strFontName.MakeUpper();

        int enclDX = PDFontGetEncodingIndex(font);
        if( -1<enclDX && enclDX<5 ){
                if(enclDX==2 &&
                        ((0<strFontName.Find("UPC")) ||
                         (0<strFontName.Find("ANGSANA")) ||
                         (0<strFontName.Find("BROWALLIA")) ||
                         (0<strFontName.Find("CORDIA")) ||
                         (0<strFontName.Find("DILLENIA")) ||
                         (0<strFontName.Find("FIXEDSYS")) ||
                         (0<strFontName.Find("MS SANS SERIF")) ||
                         (0<strFontName.Find("SYSTEM")) ) )
                        return 6;
                else
                        return 2;
        }

        ASUns8** EncArray = PDFontAcquireEncodingArray(font);
        if(EncArray!=NULL){
                CString strEnc; int i=0;

                i=0;
                strEnc=EncArray[i++];
                while( !(-1<strEnc.Find("afii")) && i < 256)
                        strEnc=EncArray[i++];
                if (i<256)
                        return 4;
        }

        AVDoc CurrentAVDoc = AVAppGetActiveDoc();
        PDDoc CurrentPDDoc = AVDocGetPDDoc(CurrentAVDoc);

        char creator[200]="", producer[200]="";

        PDDocGetInfo (CurrentPDDoc, "Creator", creator, 200);
        CString strCreator=(LPCTSTR)creator;
        strCreator.MakeUpper();
        if(0<strCreator.Find("ADOBE"))
                return 2;

        ASAtom AtomSubType = PDFontGetSubtype(font);
        CString SubType = (LPCTSTR)ASAtomGetString(AtomSubType);
        PDDocGetInfo (CurrentPDDoc, "Producer", producer, 200);
        CString strProducer=(LPCTSTR)producer;
        strProducer.MakeUpper();

        if(SubType=="TrueType"){
                if(0<strFontName.Find("MSTT")&&0<strProducer.Find("DISTILLER 3"))
                        return 3
                if( (0<strFontName.Find("UPC")&&0<strProducer.Find("DISTILLER 4")) ||
                (0<strFontName.Find("ANGSANA")&&0<strProducer.Find("DISTILLER 4")) ||
                (0<strFontName.Find("BROWALLIA")&&0<strProducer.Find("DISTILLER 4")) ||
```

```
                (0<strFontName.Find("CORDIA")&&0<strProducer.Find("DISTILLER 4")) ||
                (0<strFontName.Find("DILLENIA")&&0<strProducer.Find("DISTILLER 4")) ||
                (0<strFontName.Find("FIXEDSYS")&&0<strProducer.Find("DISTILLER 4")) ||
                (0<strFontName.Find("MS SANS SERIF")&&0<strProducer.Find("DISTILLER 4")) ||
                (0<strFontName.Find("SYSTEM")&&0<strProducer.Find("DISTILLER 4")) )
                        return 5;
        }

        if(SubType=="Type1"){
                if(0<strFontName.Find("UPC")&&0<strProducer.Find("DISTILLER 3"))
                        return 3;
                if(0<strFontName.Find("TTD"))
                        return 4;
        }

        CosObj cosObjFont = PDFontGetCosObj(font);
        CosObj FirstChKey, LastChKey;
        FirstChKey = CosDictGet(cosObjFont,ASAtomFromString("FirstChar"));
        LastChKey = CosDictGet(cosObjFont,ASAtomFromString("LastChar"));

        if (FirstChKey.b == 32 || LastChKey.b == 0 || LastChKey.b == 255)
                return 2;

        return -1;

}

void CThaiPDF::ToThaiTextNormal(PDWord word)
{
        char buff[200];

        PDWordGetString(word, buff, 200);
        gblThaiTxt += buff;
}

void CThaiPDF::ToThaiTextType3(PDWord word)
{
        ASInt16 len;
        char buff[200];

        PDWordGetString(word, buff, sizeof(buff));
        len = PDWordGetLength(word);
        for (int i = 0; i < len; i++)
                buff[i] = tblPlus35[(unsigned char)buff[i]];

        gblThaiTxt +=buff;
}

void CThaiPDF::ToThaiTextType4(PDWord word)
{
        char buff[200];
        int idxCh=0;
        CString IdxEncoding, strWord, strEnc;
        PDStyle style = PDWordGetNthCharStyle(WordFinder,word, 0);
        PDFont font = PDStyleGetFont(style);
        ASUns8** EncArray = PDFontAcquireEncodingArray(font);

        PDWordGetString(word, buff, 200);
        strWord = buff;
        for(idxCh=0;idxCh<strWord.GetLength();idxCh++) {
                strEnc = EncArray[(unsigned char)buff[idxCh]]
                if(strEnc.Left(4)=="afii" || strEnc.Left(2)=="f7")
                        strWord.SetAt(idxCh, MapGlyphName(strEnc));
        }

        gblThaiTxt += strWord;
```

```
}

void CThaiPDF::ToThaiTextType5(PDWord word)
{
	CString strWord;
	char buff[200];
	int idxCh;

	PDWordGetString(word, buff, sizeof(buff));
	strWord = buff;
	CString strTemp=strWord;
	int i=0;
	for(idxCh=0; idxCh < strTemp.GetLength(); idxCh+=2) {
		if(strTemp.GetAt(idxCh)==14)
			strWord.SetAt(i, strTemp.GetAt(idxCh+1) + 160);
		if(strTemp.GetAt(idxCh)==-9)
			strWord.SetAt(i, tblF7[(unsigned char)strTemp.GetAt(idxCh+1)]);
		i++;
	}
	if(i < strTemp.GetLength())
		strWord.SetAt(i, '\0');
	strWord.Remove(' ');

	gblThaiTxt.Insert(gblThaiTxt.GetLength(),(LPCTSTR)strWord);
}

char CThaiPDF::MapGlyphName(CString EncArrayIDX)
{
	char ThCode=32, *stopstring;

	EncArrayIDX.TrimRight(';');
	if(EncArrayIDX.Left(2)=="af") {
		int ChCode = atoi(EncArrayIDX.Right(3)) - 520;
		ThCode = tblThaiCh[(unsigned char)ChCode];
	}
	if(EncArrayIDX.Left(2)=="f7") {
		int ChCode = strtoul( EncArrayIDX.Right(2), &stopstring, 16 );
		ThCode = tblF7[(unsigned char)ChCode];
	}

	return ThCode;

}

int CThaiPDF::LookFontTbl(char fontName[])
{
	ASPathName PathName;
	char *pDIPath, inBuf[80];

	ASFile DocFile = PDDocGetFile(CurrentPDDoc);
	PathName = ASFileAcquirePathName(DocFile);
	pDIPath = ASFileSysDIPathFromPath(ASGetDefaultFileSys(), PathName, NULL);
	ASFileSysReleasePath(ASGetDefaultFileSys(), PathName);
	GetPrivateProfileString( pDIPath, fontName, "-1", inBuf, 80, "regfont.ini");
	if(inBuf==NULL)
		return -1;
	return atoi(inBuf);
}

void CThaiPDF::StoreFontTbl(int m_fontEnc, char fontName[])
{
	ASPathName PathName;
	char *pDIPath, fontEnc[2];

	ASFile DocFile = PDDocGetFile(CurrentPDDoc);
	PathName = ASFileAcquirePathName(DocFile);
	pDIPath = ASFileSysDIPathFromPath(ASGetDefaultFileSys(), PathName, NULL);
```

```
	ASFileSysReleasePath(ASGetDefaultFileSys(), PathName);
	_itoa(m_fontEnc, fontEnc, 10);

	WritePrivateProfileString( pDIPath, fontName, fontEnc, "regfont.ini" ),
	if(m_fontEnc==1)
		WritePrivateProfileString( pDIPath, "Encoding", fontEnc, "regfont.ini" );

}

CString CThaiPDF::ThaiTextExtract(PDWord word, int fontEncode)
{
	gblThaiTxt.Empty();

	//*** Convert Text from type encoding ***
		switch (fontEncode) {
		case 1:
			break;
		case 2:
			ToThaiTextNormal(word);
			break;
		case 3:
			ToThaiTextType3(word);
			break;
		case 4:
			ToThaiTextType4(word);
			break;
		case 5:
			ToThaiTextType5(word);
			break;
		default:
			ToThaiTextNormal(word);
		}
		//**************************************

	if(CancelByButton)
		gblThaiTxt.Empty();

	return gblThaiTxt;
}

int CThaiPDF::RegFont(PDWordFinder WordFinder, PDWord word, PDFont font)
{
	CFontRegDlg FontRegdlg;
	char buff[200];
	char fontName[200];
	int m_fontEncode;

	PDWordGetString(word, buff, 200);
	PDFontGetName(font, fontName, 200);
	//****** Look up Table that store Font Information ******
	m_fontEncode = LookFontTbl(fontName);
	if(-1==m_fontEncode) {                          //cann't found info then new Recognize font
		m_fontEncode = RegFontEncode(font, buff);
		if(-1<m_fontEncode)
			StoreFontTbl(m_fontEncode, fontName);
		else {
			PgressDlg.DestroyWindow();
			CString strTmp=buff;
			int i=0;
			PDWord longWord;
			while(strTmp.GetLength()<10){
				longWord = PDWordFinderGetNthWord(WordFinder, i++);
				PDWordGetString(longWord, buff, 200);
				strTmp=buff;
			}
```

```
                        for(i=0;i<strTmp.GetLength();i++){
                                strTmp.SetAt(i, tblPlus35[(unsigned char)buff[i]]);
                        }
                        strfontName = fontName;
                        strBuffType2 = buff;
                        strBuffType3 = strTmp;
                        FontRegdlg.DoModal();
                        switch (FontRegdlg.m_intFontType){
                        case 0:
                                m_fontEncode = 2;
                                break;
                        case 1:
                                m_fontEncode = 3;
                                break;
                        default :
                                m_fontEncode = 1;
                        }
                        //*****************************************
                        StoreFontTbl(m_fontEncode, fontName);
                        PgressDlg.Create(IDD_PROGRESS, NULL);
                }
        }
        return m_fontEncode;
}

bool CThaiPDF::canFindType4(PDFont font, CString pattern)
{
        char ch13, ch32;
        int foundCh13=0, foundCh32=0;
        CString IdxEncoding;

        ASUns8** EncArray = PDFontAcquireEncodingArray(font);

        IdxEncoding = EncArray[13];
        ch13 = MapGlyphName(IdxEncoding);
        foundCh13 = pattern.Find(ch13);
        IdxEncoding = EncArray[32];
        ch32 = MapGlyphName(IdxEncoding);
        foundCh32 = pattern.Find(ch32);

        if(0<foundCh13 || 0<foundCh32)
                return false;

        return true;
}

bool CThaiPDF::userConfirm(int fontEncode, PDFont font)
{
        CString strMessage;

        if(fontEncode==1){
                PgressDlg.DestroyWindow();
                strMessage.Empty();
                strMessage += "การเข้ารหัสไม่ถูกต้อง ยกเลิกการทำงาน";
                AVAlert(ALERT_STOP, (LPCTSTR)strMessage, "ตกลง", NULL, NULL, true);
                return false;
        }
        if(fontEncode==4) {
                PgressDlg.DestroyWindow();
                CString IdxEncoding;
                ASUns8** EncArray = PDFontAcquireEncodingArray(font);
                IdxEncoding = EncArray[13];
                char ch13 = MapGlyphName(IdxEncoding);
                IdxEncoding = EncArray[32];
                char ch32 = MapGlyphName(IdxEncoding);
                strMessage.Empty();
```

```
                strMessage += "เอกสารเข้ารหัสไม่ถูกต้อง\n";          strMessage += "ไม่สามารถค้นตัวอักษร ";
                strMessage += ch13;       strMessage += " และ ";      strMessage += ch32;
                strMessage += " ได้";
                AVAlert(ALERT_STOP, (LPCTSTR)strMessage, "ตกลง", NULL, NULL, true);
                return false;
        }
        if(fontEncode==5){
                PgressDlg.DestroyWindow();
                strMessage.Empty();
                strMessage += "ไม่พบการเข้ารหัสข้อความในเอกสาร\n";
                strMessage += "จะใช้การเข้ารหัสตามมาตรฐานยูนิโค้ด\n";
                strMessage += "ถ้าเอกสารมีการใช้ข้อความไทยผสมกับข้อความภาษาอังกฤษ ควรแยกกันด้วยการเว้นวรรค";
                if(1==AVAlert(ALERT_NOTE, (LPCTSTR)strMessage, "ตกลง", "ยกเลิก", NULL, true)) {
                        PgressDlg.Create(IDD_PROGRESS, NULL);
                        return true;
                }
                else
                        return false;
        }

        return true;

}

bool CThaiPDF::hasAVAlert(PDFont font)
{
        char fontName[200];
        int i=0;

        PDFontGetName(font, fontName, 200);
        while(fontList[i]!="NULL"){
                if(fontList[i] == fontName)
                        return true;
                i++;
        }

        fontList[i] = fontName;
        fontList[i+1] = "NULL";
        return false;

}
```

```
//////////////////////////////////////////////////////////////
// ProgressDlg.cpp : implementation of the CprogressDlg class
//
//////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "ThaiFind.h"
#include "ProgressDlg.h"
#include "ThaiPDF.h"
#include "FindDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////////////////////////////////////////
// CProgressDlg dialog


CProgressDlg::CProgressDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CProgressDlg::IDD, pParent)
{
	m_bCancelled	= FALSE;
	m_bModal	= FALSE;
	//{{AFX_DATA_INIT(CProgressDlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}


void CProgressDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CProgressDlg)
	DDX_Control(pDX, IDCANCEL, m_btnCancel);
	DDX_Control(pDX, IDC_PROGRESS1, m_Progress);
	//}}AFX_DATA_MAP
}


//////////////////////////////////////////////////////////////////////////
// CProgressDlg message handlers
//
//////////////////////////////////////////////////////////////////////////
BEGIN_MESSAGE_MAP(CProgressDlg, CDialog)
	//{{AFX_MSG_MAP(CProgressDlg)
	ON_WM_SHOWWINDOW()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL CProgressDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	m_Progress.SetRange(0, 100);
	m_Progress.SetPos(0);

	AVDoc CurrentAVDoc = AVAppGetActiveDoc();
	PDDoc CurrentPDDoc = AVDocGetPDDoc(CurrentAVDoc);
	TotalPage = PDDocGetNumPages(CurrentPDDoc);

	this->SetDlgItemInt(IDC_TOTAL_PAGE, TotalPage, true);

	return TRUE;  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}
```

```
void CProgressDlg::Hide()
{
   if (!::IsWindow(GetSafeHwnd()))
      return;

   if (IsWindowVisible())
   {
      ShowWindow(SW_HIDE);
      ModifyStyle(WS_VISIBLE, 0);
   }

}

void CProgressDlg::OnCancel()
{
   m_bCancelled = TRUE;
   Hide();

   if (m_bModal)
      SendMessage(WM_CLOSE);

   CWnd *pWnd = AfxGetMainWnd();
   if (pWnd && ::IsWindow(pWnd->m_hWnd))
      pWnd->SetForegroundWindow();
}

void CProgressDlg::PeekAndPump(BOOL bCancelOnESCkey /*= TRUE*/)
{
   if (m_bModal && ::GetFocus() != m_hWnd)
      SetFocus();

   MSG msg;
   while (!m_bCancelled && ::PeekMessage(&msg, NULL,0,0,PM_NOREMOVE))
   {
      if (bCancelOnESCkey && (msg.message == WM_CHAR) && (msg.wParam == VK_ESCAPE))
         OnCancel();
      // Cancel button disabled if modal, so we fake it.
      if (m_bModal && (msg.message == WM_LBUTTONUP))
      {
         CRect rect;
         m_btnCancel.GetWindowRect(rect);
         if (rect.PtInRect(msg.pt))
            OnCancel();
      }
      if (!AfxGetApp()->PumpMessage())
      {
         ::PostQuitMessage(0);
         return;
      }
   }
}
```

```
////////////////////////////////////////////////////////////
// FontRegDlg.cpp : implementation of the CFontRegDlg class
//
////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "ThaiFind.h"
#include "FontRegDlg.h"
#include "ThaiPDF.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFontRegDlg dialog
/////////////////////////////////////////////////////////////////////////////

CFontRegDlg::CFontRegDlg(CWnd* pParent /*=NULL*/)
                        : CDialog(CFontRegDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CFontRegDlg)
        m_strFontName = _T("");
        m_intFontType = 0;
        //}}AFX_DATA_INIT
}


void CFontRegDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CFontRegDlg)
        DDX_Text(pDX, IDC_FONTNAME, m_strFontName);
        DDX_Radio(pDX, IDC_MESSAGE2, m_intFontType);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CFontRegDlg, CDialog)
        //{{AFX_MSG_MAP(CFontRegDlg)
        ON_BN_CLICKED(IDC_MESSAGE2, OnRegFontType2)
        ON_BN_CLICKED(IDC_MESSAGE3, OnRegFontType3)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFontRegDlg message handlers

void CFontRegDlg::OnRegFontType2()
{
        m_intFontType = 2;
}

void CFontRegDlg::OnRegFontType3()
{
        m_intFontType = 3;
}
```

```
BOOL CFontRegDlg::OnInitDialog()
{
	CThaiPDF thaipdf;
	CDialog::OnInitDialog();

	SetDlgItemText(IDC_FONTNAME,(LPCTSTR)thaipdf.FontName);
	SetDlgItemText(IDC_MESSAGE2,(LPCTSTR)thaipdf.strType2);
	SetDlgItemText(IDC_MESSAGE3,(LPCTSTR)thaipdf.strType3);

	return TRUE,  // return TRUE unless you set the focus to a control
		// EXCEPTION: OCX Property Pages should return FALSE
}

void CFontRegDlg::OnCancel()
{
	m_intFontType = -1;

	CDialog::OnCancel();
}
```

ประวัติผู้เขียนวิทยานิพนธ์

นายสุรพงษ์ เชาว์เชี่ยวชาญ เกิดที่จังหวัดหนองคาย สำเร็จการศึกษาปริญญาตรี วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ในปีการศึกษา 2538 เคยทำงานในตำแหน่งวิศวกรระบบ แผนกคอมพิวเตอร์ บริษัท ยูทีวี จำกัด เข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2540