```
//--------------------------------------------------------------------
------
// File      : keyfob.vec
// Project   : Voice Extreme
// Purpose   : Sample program for Speaker Verification technology
// Copyright (c) 2001 by Sensory, Inc., All Rights Reserved
//--------------------------------------------------------------------
-------
//
// OPERATION:
//
//  After an initial BEEP the program loops forever, waiting for button
presses.
//  The user may want to run HyperTerminal to see debug printout from
//  the Speaker Verification process.
//
//  Button B initiates the training phase, where the program prompts for
up
//  to four passwords. The red LED is on during Pattern Generation.
//  This phase can be aborted by pressing Button A or by not responding
//  to the prompt for the next password ( it BEEPs if all are trained ).
//
//  Button A initiates a recognition phase, where the program waits for
the
//  user to say each password in order ( it BEEPs if none trained ).
//
//  Button C allows the user to cycle the "level" which controls the
tradeoff
//  between false accepts and false rejects (1-5, with 5 being the
strictest).
//
// NOTES:
//
//  This program illustrates direction of debug output to the
//  serial port and also the need to temporarily disable the RS232 lines
//  during PatGen. It also uses the #pragma directives.
//
//--------------------------------------------------------------------
-------

#include <ve.veh>             // Standard VE defs
#include "fkdemo.veh"           // SV sentence table
#include "fkdemo2.veh"          // SV sentence table

#define DEBUG_MODE      0

extern      SPEECH      VPDSNumber;
#define     BEEP  Talk(16, &VPDSNumber)

extern      WEIGHTS WTset1;
#define     CONFSET1    50

#pragma VE_APP_TEXT "Ford Keyfob Demo Program"
#pragma VE_APP_VERSION 2

#define ID0 'F'             // 3 bytes of unique ID for this sample
#define ID1 'R'
#define ID2 'D'
FLASH uint8 id[5];          // Permanent record of # of passwords
trained
```

```
#define MAX_PASSWORDS 1     // Maximum number of passwords
TEMPLATE password[MAX_PASSWORDS];
bool pwTrained;                     // is the password trainied

#define MAX_CLASSES 4       // Maximum number of SD words
TEMPLATE templates[MAX_CLASSES];
uint8 ctr;                  // Count of passwords trained

#define SV_MODE          0
#define SI_MODE          1
#define SD_MODE          2
uint8 programMode;                  // program operation mode (SV, SI or
SD)
uint8 lastProgram;

#define EASY_LEVEL   1      // easy security level
#define MEDIUM_LEVEL 3      // medium security level
#define HARD_LEVEL   5      // hardest security level
uint8 securityLevel;        // Security level
uint8 lastSecurity;

#define ARM_BTN_PRESSED       ((ReadPort1() & 0x01) > 0)
#define DISARM_BTN_PRESSED    ((ReadPort1() & 0x10) > 0)
#define PANIC_BTN_PRESSED     ((ReadPort1() & 0x02) > 0)
#define TRUNK_BTN_PRESSED     ((ReadPort1() & 0x08) > 0)

uint8 prompt;               // AskAndRecord prompt message
uint8 promptTable;                  // AskAndRecord prompt table
bool  rpMode;                       // Do simultaneous RP+Patgen
uint8 repeat;               // AskAndRecord ( 0=first try, 1=repeat )
sint8 error;                        // AskAndRecord return ( 0=success )

uint8 debugmode;              // enable debug speech

// Function Prototypes
void SetProgramMode();
void SetSecurityLevel();
void SetPassword();
void AskPassword();
sint8 AskAndRecord();
sint8 AskAndRecordSI();
void SetSDWords();
void AskSIWords();
void AskSDWords();
void ArmCar();
void DisarmCar();
void PanicMode();
void TrunkRelease();

//-------------------------------------------------------------------------
-------
main()
{
      uint8 i;

      SetDebug(SILENCE,       SPEECH_OUTPUT);
      SetDebug(PATGEN_RESULT, SPEECH_OUTPUT);
      SetDebug(Recog,         SPEECH_OUTPUT);
      SetDebug(GENERAL,       SPEECH_OUTPUT);
```

```
      debugmode = DEBUG_MODE;

    BEEP;                                // Give a welcome beep

      // configure P0.0 - P0.3 as weak inputs
      ConfigureIO(0, 0, 0);
      ConfigureIO(0, 1, 0);
      ConfigureIO(0, 2, 0);
      ConfigureIO(0, 3, 0);

      // configure P1.0, P1.1, P1.3 and P1.4 as hi-z inputs
      ConfigureIO(1, 0, 2);
      ConfigureIO(1, 1, 2);
      ConfigureIO(1, 3, 2);
      ConfigureIO(1, 4, 2);

      // configure P1.2 as a low output
      WritePort1(ReadOutputPort1() & 0xEF);
      ConfigureIO(1, 2, 3);

    SetStopCondition( PatGen, NONE );     // Set Stop Condition for
PatGen
    SetStopCondition( Talk, NONE );       // Set Stop Condition for Talk

      // set the ID bytes and get the template counts
    if ( (id[0] != ID0) || (id[1] != ID1) || (id[2] != ID2) )
    {
        id[0] = ID0;
        id[1] = ID1;
        id[2] = ID2;
        id[3] = 0;
        id[4] = 0;
    }
      pwTrained = (bool) id[3];
      ctr = id[4];

      lastProgram = 0;
      lastSecurity = 0;

    // Main loop waits for button presses and responds accordingly.
    while ( FOREVER )
    {
            SetProgramMode();

            SetSecurityLevel();

        if (DISARM_BTN_PRESSED)
        {
              switch (programMode)
          {
                case SV_MODE:     AskPassword();    break;
            case SI_MODE:     AskSIWords();     break;
                case SD_MODE:     AskSDWords();     break;
            }
                while (DISARM_BTN_PRESSED)    ;     // Wait for button
release
        }
    }
}
```

```
//-------------------------------------------------------------------------
-------
// SetProgramMode
//-------------------------------------------------------------------------
-------
void SetProgramMode()
{
uint8 i;

      i = ReadPort0() & 0x03; //DebugH8(i);
      if (i == lastProgram) return;

      do {
            lastProgram = i;
            DelayMilliSeconds(250);
            i = ReadPort0() & 0x03; //DebugH8(i);
      } while (i != lastProgram);

      switch (lastProgram)
      {
            case 0x01: programMode = SV_MODE; Talk(msg_PWDEMO,
&VPfkdemo); break;
            case 0x03: programMode = SI_MODE; Talk(msg_SIDEMO,
&VPfkdemo); break;
            case 0x02: programMode = SD_MODE; Talk(msg_SDDEMO,
&VPfkdemo); break;
      }

      if (DISARM_BTN_PRESSED)
      {
        switch (programMode)
        {
            case SV_MODE:     SetPassword();    break;
            case SI_MODE:     ;                       break;
            case SD_MODE:     SetSDWords();     break;
        }
            while (DISARM_BTN_PRESSED)    ;     // Wait for button
release
      }
}

//-------------------------------------------------------------------------
-------
// Set SetSecurityLevel
//-------------------------------------------------------------------------
-------
void SetSecurityLevel()
{
uint8 i;

      i = ReadPort0() & 0x0C; //DebugH8(i);
      if (i == lastSecurity) return;

      do {
            lastSecurity = i;
            DelayMilliSeconds(250);
            i = ReadPort0() & 0x0C; //DebugH8(i);
      } while (i != lastSecurity);
```

```
      switch (lastSecurity)
      {
            case 0x04: securityLevel = EASY_LEVEL;   Talk(msg_ERMODE,
&VPfkdemo); break;
            case 0x0C: securityLevel = MEDIUM_LEVEL; Talk(msg_MRMODE,
&VPfkdemo); break;
            case 0x08: securityLevel = HARD_LEVEL;   Talk(msg_HRMODE,
&VPfkdemo); break;
      }
}

//-------------------------------------------------------------------------
-------
// Set Password.
//-------------------------------------------------------------------------
-------
void SetPassword()
{

    do {
          prompt = msg_SYPWORD;
          promptTable = 1;
          rpMode = FALSE;
        error = AskAndRecord();     // Prompt for a password
        if (!error) {
            PutTemplate(UNKNOWN, 0, password);

                prompt = msg_REPEAT;
                promptTable = 1;
                rpMode = FALSE;
          error = AskAndRecord();     // Prompt for a password
      }
            if (!error) {
            GetTemplate(KNOWN, 0, password);
                  SetSVSecurityLevel(5);
                  error = TrainSV(UNKNOWN, KNOWN, UNKNOWN);
            }
            if (!error) {
            PutTemplate(UNKNOWN, 0, password);
            pwTrained = TRUE;
            id[3] = (uint8) pwTrained;
        }
    }
      while (error);

    Talk(msg_TCOMPLE, &VPfkdemo2);
}

//-------------------------------------------------------------------------
-------
// AskPassword
//-------------------------------------------------------------------------
-------
AskPassword()
{
    if (!pwTrained) {
            Talk(msg_PWDEMO, &VPfkdemo);
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
```

```
    }

    prompt = msg_SYPWORD;
      promptTable = 1;
    rpMode = FALSE;
    error = AskAndRecord();
    if (error) {
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
    }

      SetSVSecurityLevel(securityLevel);
      RecogSV(1, 1, 1, password);
    if (debugmode) DebugRecogSV();
      if (GetRecogSVResult() > 0) {
            Talk ( msg_PWREJEC, &VPfkdemo2);
            return;
      }

    Talk ( msg_PWACCEP, &VPfkdemo2);
    DisarmCar();
}

//-------------------------------------------------------------------------
-------
// AskAndRecord
//
// Function: Speaks prompt, does PatGen, retrying up to 3 times for
errors.
//           Aborts immediately if Button A pressed or nothing is said
for
//              first utterance.
//
// On entry: repeat = 0 for first utterance, 1 for second
//           prompt = Sentence to speak
//
// Returns:  error code from PatGen ( 0 = success )
//
//-------------------------------------------------------------------------
-------

sint8 AskAndRecord()
{
    SINT8 error;                        // PatGen error code
    UINT8 error_msg;                    // Error message
    UINT8 tries = 3;                    // 3 strikes until you're out!
    BOOL  abort = FALSE;

    do
    {
      switch (promptTable) {
            case 0: Talk(prompt, &VPfkdemo);   break;
            case 1: Talk(prompt, &VPfkdemo2);  break;
            case 2: Talk(prompt, &VPDSNumber); break;
      }

            if (!rpMode)
              error = PatGen(STANDARD);         // Record the password
            else {
```

```
              PatGen(BACKGROUND);         // Record the password
              EraseRP(ctr);
                  RecordRP(0, 0, ctr);
                  error = GetPatGenResult();
            }
            if (debugmode) DebugPatGen();

        if (error && (--tries))         // On error, if more tries
allowed
        {
            switch( error )
            {
                case INTERRUPTED:               // Abort if interrupted
                    abort = TRUE;
                    break;
                case NO_DATA:
                    if ( !repeat ) {            // Nothing said on first
try
                        abort = TRUE;           //  is an abort
condition
                        error = INTERRUPTED;
                    }
                    else                        // else just give error
                        error_msg = msg_TSOFT;
                    break;
                case TOO_LONG:
                    error_msg = msg_ERROR;
                    break;
                case TOO_NOISY:
                    error_msg = msg_ERROR;
                    break;
                case TOO_SOFT:
                    error_msg = msg_TSOFT;
                    break;
                case TOO_LOUD:
                    error_msg = msg_TLOUD;
                    break;
                case TOO_SOON:
                    error_msg = msg_ERROR;
                    break;
                default:
                    error_msg = msg_ERROR;
            }
            if (!abort)                         // Announce error unless
aborted
            {
                Talk(error_msg, &VPfkdemo2);
                DelayMilliSeconds(320);
                    prompt = msg_REPEAT;
                    promptTable = 1;
            }
        }
    // Repeat until successful, too many errors or aborted.
    } while ( error && tries && !abort );

    return error;
}

//-------------------------------------------------------------------------
-------
```

```
// AskSIWords
//-------------------------------------------------------------------------
-------
void AskSIWords()
{

uint8 siConfidence;

    prompt = 16;
      promptTable = 2;
    error = AskAndRecordSI();
    if (error) {
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
    }

    Recog(&WTset1);
      if (debugmode) DebugRecog();

    if (GetRecogMatch1() > 3) {
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
    }

      switch (securityLevel) {
            case EASY_LEVEL:   siConfidence = 0;            break;
            case MEDIUM_LEVEL: siConfidence = CONFSET1;     break;
            case HARD_LEVEL:   siConfidence = 90;           break;
      }
      if (GetRecogLevel1() < siConfidence) {
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
    }

      switch (GetRecogMatch1())
      {
            case 0: {
              Talk(msg_ARM, &VPfkdemo2);
                  ArmCar();
                  break;
            }
            case 1: {
            Talk(msg_DISARM, &VPfkdemo2);
                  DisarmCar();
                  break;
            }
            case 2: {
            Talk(msg_PMODE, &VPfkdemo2);
                  PanicMode();
                  break;
            }
            case 3: {
            Talk(msg_TRELEAS, &VPfkdemo2);
                  TrunkRelease();
                  break;
            }
    }
```

```
}

//-------------------------------------------------------------------------
-------
// AskAndRecordSI
//
// Function: Speaks prompt, does PatGen, retrying up to 3 times for
errors.
//           Aborts immediately if Button A pressed or nothing is said
for
//              first utterance.
//
// On entry: repeat = 0 for first utterance, 1 for second
//           prompt = Sentence to speak
//
// Returns:  error code from PatGen ( 0 = success )
//
//-------------------------------------------------------------------------
-------

sint8 AskAndRecordSI()
{
    SINT8 error;                        // PatGen error code
    UINT8 error_msg;                    // Error message
    UINT8 tries = 3;                    // 3 strikes until you're out!
    BOOL  abort = FALSE;

    do
    {
      switch (promptTable) {
            case 0: Talk(prompt, &VPfkdemo);   break;
            case 1: Talk(prompt, &VPfkdemo2);  break;
            case 2: Talk(prompt, &VPDSNumber); break;
      }

        error = PatGenW(STANDARD, &WTset1);           // Record the
password
            if (debugmode) DebugPatGen();

        if (error && (--tries))         // On error, if more tries
allowed
        {
            switch( error )
            {
                case INTERRUPTED:               // Abort if interrupted
                    abort = TRUE;
                    break;
                case NO_DATA:
                    if ( !repeat ) {            // Nothing said on first
try
                        abort = TRUE;           //  is an abort
condition
                        error = INTERRUPTED;
                    }
                    else                        // else just give error
                        error_msg = msg_TSOFT;
                    break;
                case TOO_LONG:
                    error_msg = msg_ERROR;
                    break;
```

```
                case TOO_NOISY:
                    error_msg = msg_ERROR;
                    break;
                case TOO_SOFT:
                    error_msg = msg_TSOFT;
                    break;
                case TOO_LOUD:
                    error_msg = msg_TLOUD;
                    break;
                case TOO_SOON:
                    error_msg = msg_ERROR;
                    break;
                default:
                    error_msg = msg_ERROR;
            }
            if (!abort)                         // Announce error unless
aborted
            {
                Talk(error_msg, &VPfkdemo2);
                DelayMilliSeconds(320);
                    prompt = msg_REPEAT;
                    promptTable = 1;
            }
        }
    // Repeat until successful, too many errors or aborted.
    } while ( error && tries && !abort );

    return error;
}

//-------------------------------------------------------------------------
-------
// Set SetSDWords
//-------------------------------------------------------------------------
-------
void SetSDWords()
{
      for (ctr = 0; ctr < MAX_CLASSES; ctr++) {
          do {
            switch (ctr) {
                  case 0: prompt = msg_SFCOMMA; promptTable = 0; break;
                  case 1: prompt = msg_SSCOMMA; promptTable = 0; break;
                  case 2: prompt = msg_STCOMMA; promptTable = 0; break;
                  case 3: prompt = msg_SFOCOMM; promptTable = 1; break;
                  }
                rpMode = FALSE;
             error = AskAndRecord();     // Prompt for a password
          if (!error) {
                  PutTemplate(UNKNOWN, ctr, templates);

                      prompt = msg_REPEAT;
                      promptTable = 1;
                      rpMode = TRUE;
                error = AskAndRecord();     // Prompt for a password
            }
                  if (!error) {
              GetTemplate(KNOWN, ctr, templates);
              SetSDPerformance(5);
                error = TrainSD(UNKNOWN, KNOWN, UNKNOWN);
                  }
```

```
            if (!error) {
                PutTemplate(UNKNOWN, ctr, templates);
                if (ctr > 0) {
                              SetSDPerformance(securityLevel);
                              RecogSD(ctr, templates);
                              if (debugmode) DebugRecogSD();
                              error = (GetRecogSDResult() == 0);
                              if (error) {
                              Talk(msg_TSTO, &VPfkdemo2);
                              DelayMilliSeconds(250);
                                    PlayRP(GetRecogSDClass1());
                              }
                    }
                    }
            if (!error) {
                  id[4] = ctr+1;
                  PostRP(ctr);
          }
          }
            while (error);
      }

    Talk(msg_TCOMPLE, &VPfkdemo2);
}

//-------------------------------------------------------------------------
-------
// AskSDWords
//-------------------------------------------------------------------------
-------
void AskSDWords()
{
    if (ctr< MAX_CLASSES) {
            Talk(msg_SDDEMO, &VPfkdemo);
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
    }

    prompt = 16;
      promptTable = 2;
      rpMode = FALSE;
    error = AskAndRecord();
    if (error) {
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
      return;
    }

      SetSDPerformance(securityLevel);
    RecogSD(MAX_CLASSES, templates);
    if (debugmode) DebugRecogSD();

      if (GetRecogSDResult() > 0) {
            Talk(msg_ERROR, &VPfkdemo2);
      BEEP;
            return;
      }

      switch (GetRecogSDClass1())
```

```
      {
            case 0: {
            PlayRP(0);
                  ArmCar();
                  break;
            }
            case 1: {
            PlayRP(1);
                  DisarmCar();
                  break;
            }
            case 2: {
            PlayRP(2);
                  PanicMode();
                  break;
            }
            case 3: {
            PlayRP(3);
                  TrunkRelease();
                  break;
            }
    }
}

//-------------------------------------------------------------------------
-------
// ArmCar
//-------------------------------------------------------------------------
-------
void ArmCar()
{
      // bring P1.0 high for 1 second
      WritePort1(ReadOutputPort1() | 0x01);
      ConfigureIO(1, 0, 3);
      DelaySeconds(1);
      ConfigureIO(1, 0, 2);
      BEEP;
}

//-------------------------------------------------------------------------
-------
// DisarmCar
//-------------------------------------------------------------------------
-------
void DisarmCar()
{
      // bring P1.2 high for 1 second
      WritePort1(ReadOutputPort1() | 0x04);
      DelaySeconds(1);
      WritePort1(ReadOutputPort1() & 0xFB);
      BEEP;
}

//-------------------------------------------------------------------------
-------
// PanicMode
//-------------------------------------------------------------------------
-------
void PanicMode()
{
```

```
      // bring P1.1 high for 1 second
      WritePort1(ReadOutputPort1() | 0x04);
      ConfigureIO(1, 1, 3);
      DelaySeconds(1);
      ConfigureIO(1, 1, 2);
      BEEP;
}

//-------------------------------------------------------------------------
-------
// TrunkRelease
//-------------------------------------------------------------------------
-------
void TrunkRelease()
{
      // bring P1.3 high for 1 second
      WritePort1(ReadOutputPort1() | 0x08);
      ConfigureIO(1, 3, 3);
      DelaySeconds(1);
      ConfigureIO(1, 3, 2);
      BEEP;
}
```