

# 1. Introduction

This information note describes how to incorporate FOUNDATION™ Digital field programmable products into fitting software. It is intended for those who already have an understanding of programming with ARK. For an introduction to ARK, refer to the [ARK Tutorial](#) (Document #12248).

---

**Note:** Code snippets written in Visual Basic 6 are included for illustration purposes. Integration using other programming languages should be similar.

---

## 2. IFoundationProduct3 Interface

### 2.1 ACCESSING IFoundationProduct3

To access the specific product methods and properties for FOUNDATION Digital, it is necessary to create an IFoundationProduct3 interface for the selected product. This can be done through the GetFoundationProduct function as follows:

```
Function GetFoundationProduct(prod As IProduct) As IFoundationProduct3
    If prod.ControllerInfo.Name = "GC5020" Then
        Set GetFoundationProduct = prod
    Else
        `The selected product is not a FOUNDATION Digital
        Set GetFoundationProduct = Nothing
    End If
End Function
```

### 2.2 IFoundationProduct3 Methods

There are several properties and methods exposed through the IFoundationProduct3 interface. [Table 1](#) describes methods that are essential to incorporating FOUNDATION Digital into fitting software.

**Table 1: Key IFoundationProduct3 Properties and Methods**

Method or Property	Description
IsFieldProduct	Indicates that the product is a field product
IsLocked	Indicates that the connected device is locked to a configuration
NumberOfMemories	The number of memories for the chosen product

## 3. Communication

Communication with FOUNDATION Digital field programmable products is designed to only allow the manipulation of up to eight resistor values. Therefore, initialization of the device, reading from the device, and burning to the device are performed differently than for other Sound Design Technologies' products.

### 3.1 Initialization

Reads and burns only access the resistor values so that communication speed can be very high. This means that the other non-field programmable parameters must be read in from the device on initialization. This can be accomplished using the following initialization routine:

```
Sub Init()  
    Dim i As Integer  
    Dim control As Memory  
    Dim FoundProd As IFoundationProduct3  
    Dim mem As IMemory2  
  
    'Standard initialization  
    Set control = m_prod.GetFull(fControl)  
    m_prog.InitEx CURRENT_EAR, control  
    m_prod.SetControllerMemoryEx control  
  
    'Obtain the FOUNDATION interface  
    Set FoundProd = GetFoundationProduct(m_prod)  
  
    'Make sure that the device is locked.  
    If Not FoundProd.IsLocked Then  
        MsgBox "The connected device will not work with this application."  
        Exit Sub  
    End If  
  
    'If the product is a FOUNDATION product  
    If Not FoundProd Is Nothing Then  
        'Read each of the audiological memories  
        For i = 0 To FoundProd.NumberOfMemories - 1  
            Set mem = m_prod.GetFull(fResistor)  
            Set m_Cached.Memories(i) = mem  
            m_prog.ReadEx CURRENT_EAR, i,  
                m_Cached.Memories(i)  
  
            'Set the memory to the product if it is from the current memory  
            If CURRENT_MEMORY = i Then  
                m_prod.SetControllerMemoryEx  
                    m_Cached.Memories(i)  
            End If  
        Next i  
    End If  
End Sub
```

The `m_Cached.Memories()` variable is of type `IMemory` or `IMemory2`. It should be maintained for the duration of communication so that when memory changes take place, the entire audiological memory can be assigned to the product and displayed to the screen without having to perform a full read from the device once again.

The connected FOUNDATION Digital device must also be queried to determine if it has been locked. An unlocked device does not have a valid configuration and cannot be used with a field product. This ensures that the device has been locked to a configuration and is compatible with a field product.

## 3.2 Reading

As all audiological parameters are read in at initialization and only resistor settings can change, subsequent reads from the device only update the resistor settings. This is done through the `Read` method from the programmer interface. Because incremental burns are not possible with field products, the memory object read from the device needs to be assigned to the product using the `SetControllerMemoryEx` method, but does not need to be maintained, as illustrated in the following `ReadDevice` routine:

```
Sub ReadDevice()  
    Dim FoundProd As IFoundationProduct3  
    Dim IsFieldProduct As Boolean  
    Dim mem As IMemory2  
  
    'Obtain the FOUNDATION interface  
    Set FoundProd = GetFoundationProduct(m_prod)  
  
    'Determine if the product is a field product  
    IsFieldProduct = False  
    If Not FoundProd Is Nothing Then  
        IsFieldProduct = FoundProd.IsFieldProduct  
    End If  
  
    If IsFieldProduct Then  
        'Read the resistor values for the memory (i.e. R1-R8)  
        Set mem = m_prog.Read(CURRENT_EAR, CURRENT_MEMORY)  
        m_prod.SetControllerMemoryEx mem  
    Else  
        'Perform reading for other products as per usual  
    End If  
End Sub
```

### 3.3 Burning

Burning to FOUNDATION Digital field programmable products requires that the memory object be retrieved from the device, burned to the device, and then written to the registers, as illustrated in the following BurnDevice routine:

```
Sub BurnDevice()  
    Dim FoundProd As IFoundationProduct3  
    Dim IsFieldProduct As Boolean  
    Dim mem As IMemory2  
  
    'Obtain the FOUNDATION interface  
    Set FoundProd = GetFoundationProduct(m_prod)  
  
    'Determine if the product is a field product  
    IsFieldProduct = False  
    If Not FoundProd Is Nothing Then  
        IsFieldProduct = FoundProd.IsFieldProduct  
    End If  
  
    If IsFieldProduct Then  
        'Get the memory object  
        Set mem = m_prod.GetControllerMemoryEx(From, fResistor, Nothing)  
    Else  
        'Retrieve memory objects for other products as per usual  
    End If  
  
    'Set the valid field if it exists  
    If ValidExists(mem) Then  
        mem("Valid").Value = VALID_STATE  
    End If  
  
    'Burn the memory to the device  
    m_prog.Burn CURRENT_EAR, CURRENT_MEMORY,  
        mem, VERIFY_MEMORY  
  
    'Write the current settings to the device if it is a field product  
    If IsFieldProduct Then  
        m_prog.Write CURRENT_EAR, CURRENT_MEMORY, mem  
    End If  
End Sub
```

### 3.4 Valid Bit

The number of memories exposed in any FOUNDATION Digital product is defined when the product is created through ARKonline™. As a result, the valid bit is not available and the number of memories can be accessed with the NumberOfMemories property.

Because other products require the valid bit to be set accordingly, it must be determined if a product has a valid bit before an assignment can be made. This can be done using the following routine:

```
Function ValidExists(mem As IMemory) As Boolean
    On Error GoTo ValidDoesntExist
    `Assume valid does not exist
    ValidExists = False
    `Check if valid exists
    If Not mem.Parameters("Valid") Is Nothing Then
        ValidExists = True
    End If
    Exit Function
ValidDoesntExist:
End Function
```

## 4. Revision History

Version	ECR	Date	Change Description
1	148594	March 2008	Document conversion to new template and editing.

**CAUTION**

ELECTROSTATIC SENSITIVE DEVICES  
DO NOT OPEN PACKAGES OR HANDLE  
EXCEPT AT A STATIC-FREE WORKSTATION



DOCUMENT IDENTIFICATION

**INFORMATION NOTE**

Information relating to this product and the application or design described herein is believed to be reliable, however such information is provided as a guide only and Sound Design Technologies assumes no liability for any errors in this document, or for the application or design described herein. Sound Design Technologies reserves the right to make changes to the product or this document at any time without notice.

**SOUND DESIGN TECHNOLOGIES**

Mailing Address: P.O. Box 278 , Burlington , Ontario , Canada , L7R 3Y2

Sound Design Technologies assumes no liability for any errors or omissions in this document, or for the use of the circuits or devices described herein. The sale of the circuit or device described herein does not imply any patent license, and Sound Design Technologies makes no representation that the circuit or device is free from patent infringement.

Sound Design Technologies and Sound Design Technologies logo are registered trademarks of Sound Design Technologies, Ltd.

© Copyright 2007 Sound Design Technologies, Ltd. All rights reserved. Printed in Canada.

[www.SoundDesignTechnologies.com](http://www.SoundDesignTechnologies.com)