



# Intel Mobile Platform Hospital IT Enabling Research

Software and Solutions Group

*Highland Mary Mountain, Sr. SW Engineer/Architect  
SSG/GDRD/Digital Health Enabling*

*May 7, 2008*

[Mark] is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States or other countries



# Agenda

- What does my group do?
  - SSG – Developer Relations Division
  - Digital Health Enabling
- RFID Application in Hospital IT
- RFID Reader API – What is the value – relating to HIT?
- Intel Mobile Platform SDK
- RFID Reader API – What is it?
- Summary of R&D Discoveries
- Backup
  - Lab Code Business Context
  - RFID Reader and Tags definitions
  - Lab Code Scenarios

[Mark] is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States or other countries



# What does my group do?

- Work with Healthcare ISVs to get their applications to run best on Intel. (GE, McKesson, Cerner etc)
- R&D for Mobile Point of Care solutions (aka Mobile Clinician's Assistant – Motion C5 Tablet)
  
- This presentation will cover the highlights of the MPOC R&D activities over the last 2+ years.

# Patient Identification Issues in healthcare

- **Patient identification issues in healthcare facilities (as presented by Zebra\* Feb 2005)**
- In most healthcare facilities there is no patient identification technology. Handwritten wristbands have problems such as:
  - 8.6% of them contain incorrect data
  - 5.7% are illegible to some degree
  - Mismatching patients to their care can involve
    - Giving the wrong medication
    - Wrong blood type administered
    - Pathology samples mixed up
    - Wrong part of body operated on/ removed during surgery
- The solution is Positive Patient ID which involves tracking the patient at every stage of their treatment ie:
  - Medication administration
  - Blood administration
  - Sample management
  - Treatment – Xray, surgical etc
  - Non medical such as meal orders and patient billing

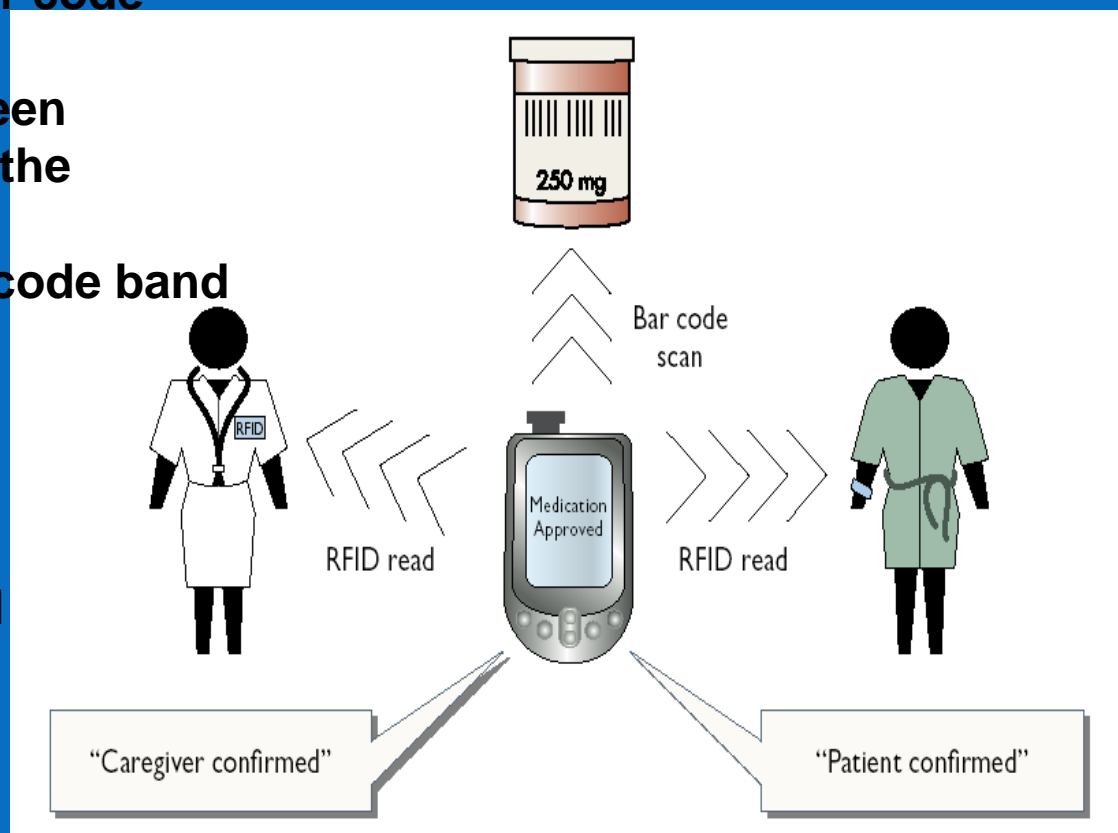
# RFID and Healthcare IT

## Barcodes are not going away...but...

**\*Barcode for patient added 2.5 seconds to each medication administration event**

- Uncover patient's wristband
- Rotate the wristband's bar code into view
- Ensure line of sight between scanner and barcode Aim the scanner
- Damaged or wrinkled barcode band
- Causes problems

**The opportunity is there to introduce new technology in HIT, but within limited Hospital IT budgets!**



# RFID vs. Bar Codes

## RFID

- + Identify an individual item
- + Can be scanned automatically
- + Only need to be within reach of reader
- + More robust
- + Can read multiple tags at once  
(bulk reading)
- Fifty cents per tag – but coming down to five cents

## Bar Codes

- Identify a product group
- Need to be scanned manually
- Need to be in line of sight
- Easily damaged
- Can only read one by one
- + Cheap – a penny a tag

**RFID is much more efficient than Bar Codes!**

# RFID Passive Tag Variations SW Development Challenges

- “The great thing about standards, is that there are so many of them”
  - Many passive RFID Tag Standards
  - Various Manufactures
  - Many (undocumented) variations across Manufactures
- HF vs. UHF
  - HF established in near field
  - UHF attempting to break into near field
  - Healthcare favors HF, but UHF trying to gain traction
- Costs comparisons
  - Generally speaking, UHF Gen2 Tags are substantially less expensive than HF tags

# RFID Tags – Memory Structures

## Class 1 Generation 2

### K.2 Tag memory contents and lock-field values

Table K.1 and Table K.2 show the example Tag memory contents and lock-field values, respectively.

Table K.1 — Tag memory contents

Memory Bank	Memory Contents	Memory Addresses	Memory Values
TID	TID[15:0]	10 <sub>h</sub> –1F <sub>h</sub>	54E2 <sub>h</sub>
	TID[31:16]	00 <sub>h</sub> –0F <sub>h</sub>	A986 <sub>h</sub>
EPC	EPC[15:0]	50 <sub>h</sub> –5F <sub>h</sub>	3210 <sub>h</sub>
	EPC[31:16]	40 <sub>h</sub> –4F <sub>h</sub>	7854 <sub>h</sub>
	EPC[47:32]	30 <sub>h</sub> –3F <sub>h</sub>	BA08 <sub>h</sub>
	EPC[63:48]	20 <sub>h</sub> –2F <sub>h</sub>	FEDC <sub>h</sub>
	PC[15:0]	10 <sub>h</sub> –1F <sub>h</sub>	2000 <sub>h</sub>
	CRC-16[15:0]	00 <sub>h</sub> –0F <sub>h</sub>	as calculated (see <a href="#">Annex E</a> )
Reserved	access password[15:0]	30 <sub>h</sub> –3F <sub>h</sub>	00DE <sub>h</sub>
	access password[31:16]	20 <sub>h</sub> –2F <sub>h</sub>	ACCE <sub>h</sub>
	kill password[15:0]	10 <sub>h</sub> –1F <sub>h</sub>	00DE <sub>h</sub>
	kill password[31:16]	00 <sub>h</sub> –0F <sub>h</sub>	DEAD <sub>h</sub>

Table K.2 — Lock-field values

Kill Password		Access Password		EPC Memory		TID Memory		User Memory	
1	0	1	0	0	0	0	0	N/A	N/A

# RFID Reader Abstraction Value

## ReadUserData: Serial Command Interface

### 3.6.4 Example 4: Binary Mode (Tagit HF)

Request :

	Message Length	Flags	Command	Tag Type	TID	Starting Block	Number of Blocks	CRC
<STX>	0x0B	0x60	0x24	0x03	0x01321FA7	0x07	0x01	0x6C23

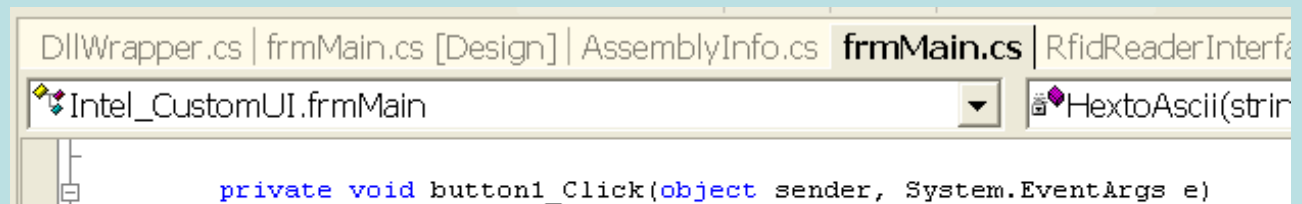
Read 0x01 block of a Tag-It HF tag starting from the address 0x07.

Response :

	Message Length	Response Code	Data	CRC
<STX>	0x07	0x24	0x11223344	0xA023

The operation is successful and the data is returned back to the host.

## ReadUserData: .Net\* w/ IMPSDK RFIDReader



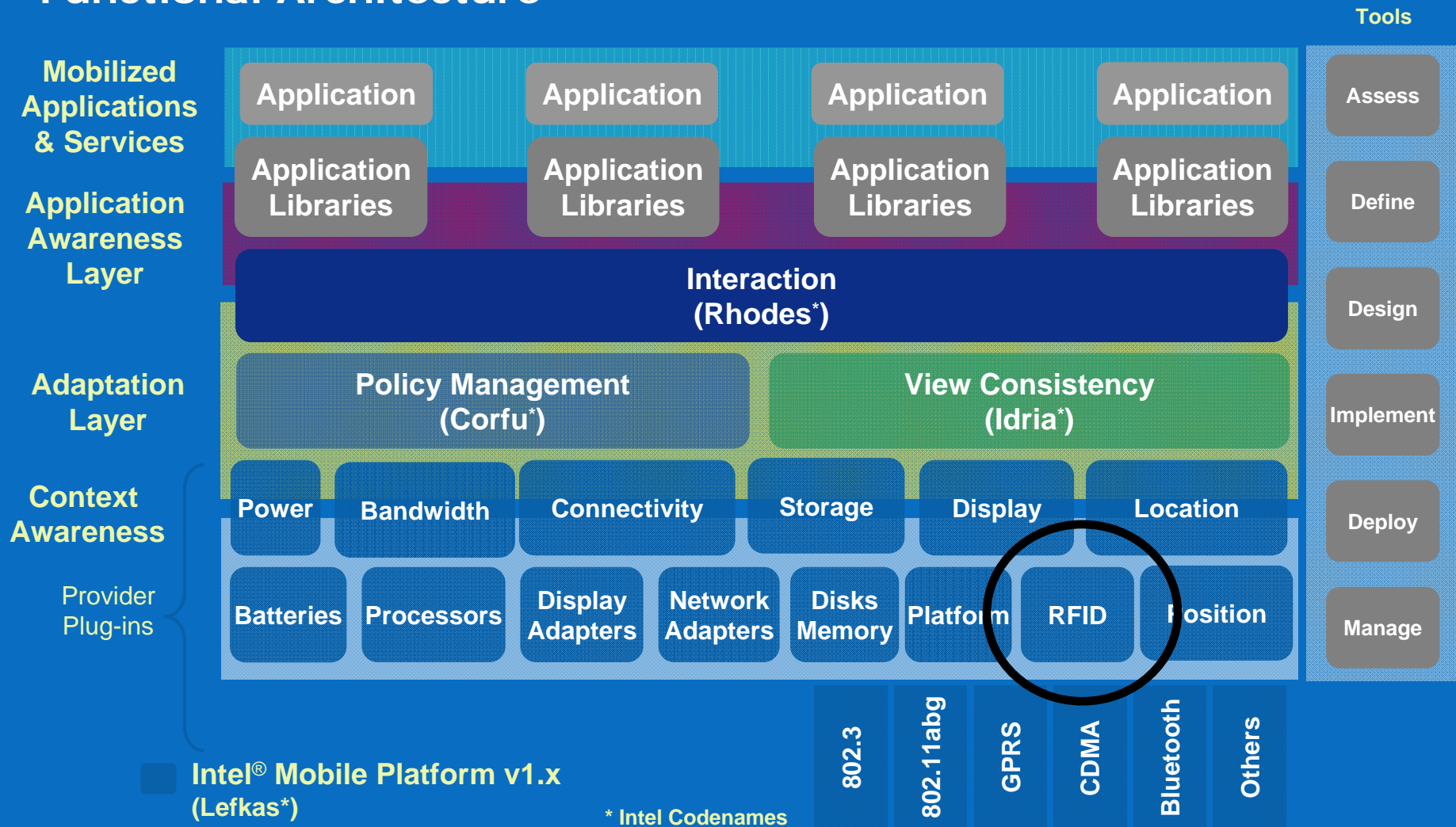
```
DllWrapper.cs | frmMain.cs [Design] | AssemblyInfo.cs | frmMain.cs | RfidReaderInterfa
Intel_CustomUI.frmMain HextoAscii(strin
private void button1_Click(object sender, System.EventArgs e)
```

For more Information Attend Lab:  
"New features in the Intel® Mobile Platform SDK v1.2 (Lab)"  
Wednesday@ 2:00 pm & 3:45 in Garden 4



# Intel® Mobile Platform SDK

## Functional Architecture



# Goals of RFID Reader Object

- Goals
  - Provide high level abstraction of basic RFID Reader - Tag Operations
  - Tool non-device driver developers with Reader Operations methods
  - Ease of integration for .Net\*/Java\* based UI
- Non-goals
  - Fully exhaustive API for all possible RFID Reader/Tag Type combinations
- Usage Models
  - Basic ID Scan (Single Tag)
  - Continuous Scanning Mode
  - Read Tag Data (User Data)
  - Write Tag Data (User Data)
  - Write Tag ID and EPC (Electronic Product Code)
  - Erase, Kill, Lock, etc.

**For more Information Attend Lab:**  
"New features in the Intel® Mobile Platform SDK v1.2 (Lab)"  
Wednesday@ 2:00 pm & 3:45 in Garden 4



# The Intel® Mobile Platform SDK

- RFID Reader Class API

Object	Properties	Methods	Events
<b>RFIDReader</b>	<b>Manufacturer Name</b> <b>SerialNumber</b> <b>FirmwareRevision</b> <b>ConnectionInterface</b> <b>BaudRate</b> <b>SerialPort</b> <b>Protocols</b> <b>SupportedProtocols</b> <b>Enabled</b> <b>PowerLevel</b> <b>OperatingFrequencies</b>	StartReadingTags() StopReadingTags() ReadTags() ReadUserData() WriteUserData() EraseUserData() WriteTagId() ReadEpc() WriteEpc() ReadTagField() WriteTagField() LockTag() PermanentlyLockTag() UnlockTag() PermanentlyUnlockTag() KillTag() SetTagPassword() RemoveTagPassword() ChangeTagPassword() QuietTag()	<b>TagFound</b> <b>DeviceAvailable</b>

**RFID Reader Device Abstraction**

# SSG's AE Mobile Point of Care Investigation

- Strategy #1: Context Aware Computing via patient/clinician/physician tracking
  - Real Time Location (RTLS) – enable tracking the device
  - ISV's said: Too difficult and expensive – still on path to maturity
- Strategy #2: RFID tags will replace barcodes enable integration via simple API
  - Barcodes are here to stay
  - ISV's said: nice Mobile Platform SDK RFID Reader API, we will use it at some point
  - ISV's said: Exposing platform state is what we really want (Battery power, connectivity, bandwidth indicators for end users)

All along the **pink elephant** in the corner was...

- ISV legacy UIs won't be usable on these small devices
- Legacy Applications drive ISV revenue
- Deploying SSF devices is a secondary priority when allocating ISV development resources





# RFID vs. RTLS

## RFID – Radio Frequency Identification

1. Can determine location only when tagged object is in the range of an RFID Reader.
2. Also called Passive RFID to denote the passive “tags” being used
3. Passive RFID Tags require an RFID Reader to “excite” or power the tags since these tags have no power source and are dependent on the RFID Readers to operate.
4. RFID Tag Protocols - Passive Tags and associated RFID Readers operate together via standard tag protocols (or air protocols). These protocols specify what is communicated over the “air”, what is stored on the tag, etc.

## RTLS – Real-time Location Systems

1. The system actively tracks the position of a tracked object at any point in time.
2. WiFi 802.11 RTLS is sometimes categorized as an RFID technology, but it really is not.
3. Active Tags are small self powered devices. This project will use WiFi (802.11) enabled active tags.
4. 802.11 Protocol is the common protocol in wireless networks used in Hospitals. Wi-Fi 802.11 based RTLS will be used for this project. Any 802.11 enabled device can be tracked by this RTLS technology, including laptops and PDAs.



# Lab Code Business Context

- Digital Health Use Case
  - 5 R's (right medication, right dose, right route, right time, right patient – “the right medicine in the right dose by the right route at the right time gets to the right patient”).
- Application Development Environment
  - Visual Studio, C#, Intel Mobile Platform SDK RFID Provider

**App Developer Shielded from RFID Reader Complexities**



# Healthcare Prototype


The screenshot shows a window titled "Patient Info" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains the following fields and controls:

- Patient ID:** A text input field with a "Scan Tag" button to its right.
- First Name:** A text input field.
- Last Name:** A text input field.
- SSN:** A text input field.
- Blood Type:** A dropdown menu with the text "Select Blood Type" and a downward arrow.
- Patient Picture:** A text label with a "Select Image" button to its right.
- Update Patient:** A button centered at the bottom of the window.


# Healthcare Prototype

**CreateOrder**

Scan Patient



FirstName	<b>Randy</b>
RfidTagID	<b>E00400001F138D01</b>
SSN	<b>111-11-1111</b>
LastName	<b>Miller</b>
ImagePath	<b>C:\RFID CVS\RFIDPOC\MRTE</b>

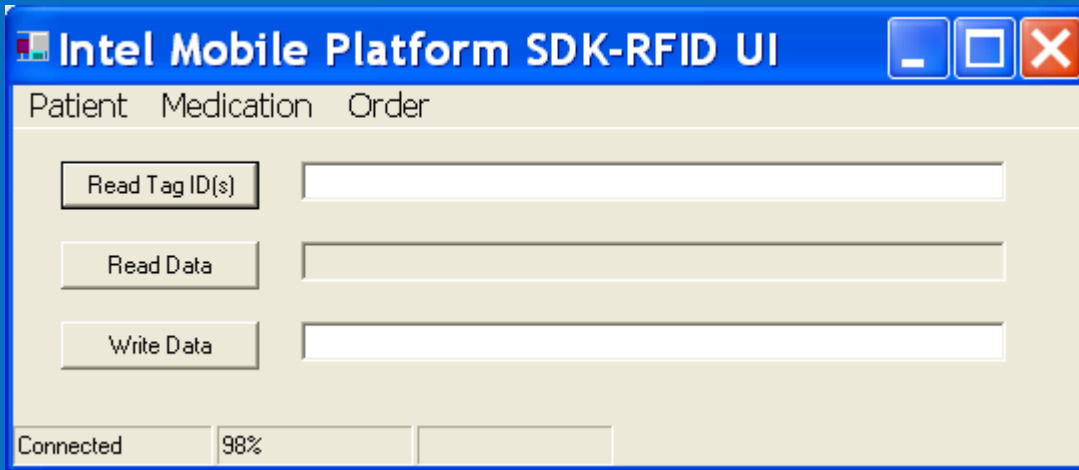


**Medication List**

	Dosage	RfidTagID	Name	ActiveIngredi	Manufacturer	GenericName	ImagePath
▶	2 sprays in ea	300833B2DD	Flonase	fluticasone	Bayer Corpor	fluticasone na	C:\RFID CVS
	1 tablet 2 tim	72616E6479	Zyrtec	Pseudoephed	Zila, Inc.	Cetirizine	C:\RFID CVS

Create Order

# Lab Exercise



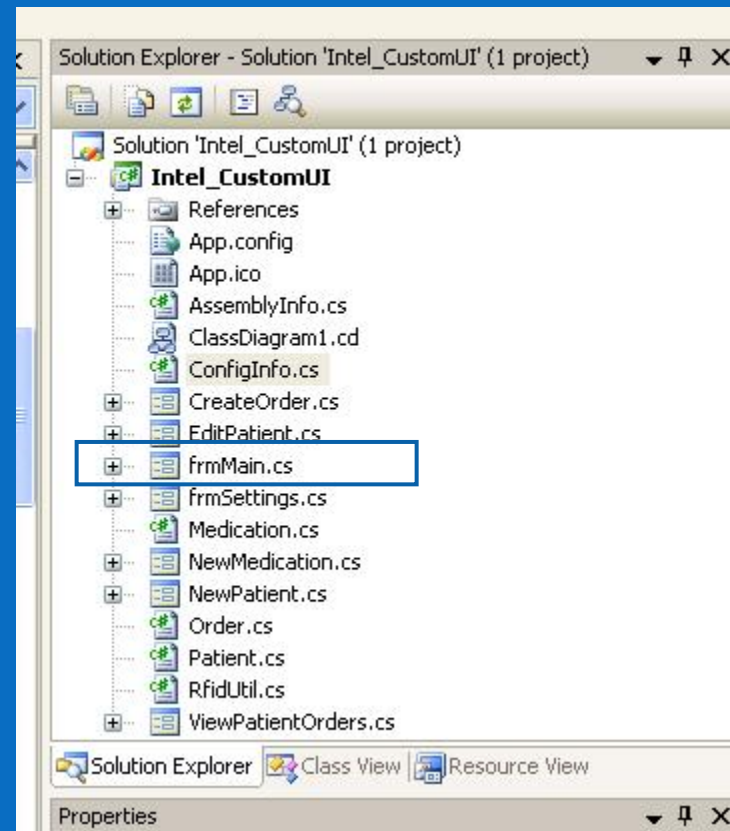
# RFID Reader and Tags Equipment (pg 3)

- SkyeTek M1
  - HF Reader Development Platform
- RFID HF Tags
  - HF Badges...Nurse Badge for Authentication
  - HF Wristband-like Tags....for Patient Identification
  - HF Item Level Tags



Many components go into an RFID Enabled Application

# C# Project



# Lab Exercises

Scan for Tag IDs in RF field

Intel Mobile Platform SDK - RFID UI

Patient Medication Order

Read Tag ID(s)

Read Data

Write Data

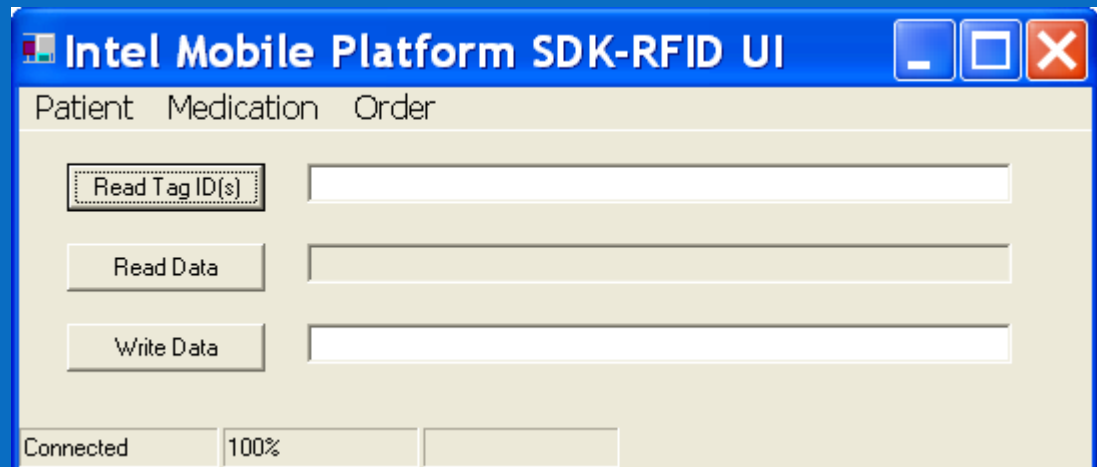
Connected 100% Skyetek : M1

Read User Data for Tag in RF field

Write User Data for Tag in RF field

Populate Form Status Bar with RFID Reader property info

# frmMain.cs - frmMain\_Load



```
private void frmMain_Load(object sender, System.EventArgs e)
{
    InitReader();

    StatusBarPanel networkStatus = new StatusBarPanel();
    statusBar1.Panels.Add(networkStatus);

    StatusBarPanel batteryStatus = new StatusBarPanel();
    statusBar1.Panels.Add(batteryStatus);

    ContextClass myClass = new ContextClass();

    ConnectivityInstance myInstance = (ConnectivityInstance) myClass.GetInstance("Connectivity");
    PowerInstance powInstance = (PowerInstance) myClass.GetInstance("Power");

    networkStatus.Text = myInstance.IsConnected.GetValue() ? "Connected" : "Disconnected";
    batteryStatus.Text = powInstance.PercentRemaining.GetValue().ToString() + "%";
}
```

# Exercise #1: frmMain.cs – InitReader()

```
/// <summary>
/// TODO Exercise 1:   Instantiate the IMPSDK RFID Reader Instance in .Net C#
///                   Uncomment code within function to establish reader instance
///                   InitReader()
/// </summary>

private void InitReader()
{

    RfidReaderClass readerClass = new RfidReaderClass();
    System.Threading.Thread.Sleep(3000); //buf with serial port manager
    RfidReaderCollection readerCollection = (RfidReaderCollection)readerClass.GetInstances();

    if(!readerCollection.HasNext())
    {
        MessageBox.Show("No RFID Readers found");
    }
    else
    {
        reader = (RfidReaderInstance)readerCollection.Next();
    }

}
```

# Exercise #2:

## frmMain.cs – InitReaderStatus()

```
/// <summary>
/// TODO Exercise 2:   Attain RFID Reader Property Information for Status Bar
/// </summary>
/// <param name="rfidStatus"></param>

private void InitReaderStatus(StatusBarPanel rfidStatus)
{

    if(!reader.Open())
    {
        MessageBox.Show("Failed to open RFID Reader connection");
    }
    else
    {

        rfidStatus.Text = reader.Manufacturer.GetValue() + " : " + reader.Name.GetValue();
        rfidStatus.ToolTipText = "BaudRate = " + reader.BaudRate.GetValue().ToString() + "\nSerial Port = "
            + reader.SerialPort.GetValue().ToString() + "\nSerial Number = " + reader.SerialNumber.GetValue();

        reader.Close();
    }
}
```

# Exercise #3:

## frmMain.cs – btnReadTagIDs\_Click()

```
/// <summary>
/// TODO Exercise 3:   Read Tag Ids and show them in first text box
///                   Uncomment code and replace <student input> with lesson content
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void btnReadTagIDs_Click(object sender, System.EventArgs e)
{
    // C# code specific to form and textbox operation
    this.Cursor = Cursors.WaitCursor;

    if(!reader.Open())
    {
        MessageBox.Show("Failed to open RFID Reader connection");
    }
    else
    {
        ArrayList tags = new ArrayList();
        if(!reader.ReadTags(5000, tags))
        {
            MessageBox.Show("ReadTags method failed");
        }

        foreach(TagInfo ti in tags)
        {
            if(txtReadTagIDs.Text == "")
            {
                txtReadTagIDs.Text = ti.TagId;
            }
            else
            {
                txtReadTagIDs.Text = txtReadTagIDs.Text + " : " + ti.TagId;
            }
        }

        reader.Close();
    }

    // C# code specific to form and textbox operation
    this.Cursor = Cursors.Default;
}
}
```

# Exercise #4:

## frmMain.cs – btnWriteUserData\_Click()

```
/// <summary>
/// TODO Exercise 4:   Write User Data - Same Wristband form factor tag
///                   Uncomment code and replace <student input> with lesson content
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void btnWriteUserData_Click(object sender, System.EventArgs e)
{
    // C# code specific to form and textbox operation
    this.Cursor = Cursors.WaitCursor;

    if(!reader.Open())
    {
        MessageBox.Show("Failed to open RFID Reader connection");
    }
    else
    {
        ArrayList data = StringToByteArrayList(txtWriteUserData.Text);
        if(!reader.WriteUserData(data))
        {
            MessageBox.Show("WriteUserData method failed");
        }

        reader.Close();
    }

    // C# code specific to form and textbox operation
    this.Cursor = Cursors.Default;
}
```

# Exercise #5:

## frmMain.cs – btnReadUserData\_Click()

```
/// <summary>
/// TODO Exercise 5:   Read User Data - Wristband form factor tag
///                   Uncomment code and replace <student input> with lesson content
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void btnReadUserData_Click(object sender, System.EventArgs e)
{
    // C# code specific to form and textbox operation
    this.Cursor = Cursors.WaitCursor;

    if(!reader.Open())
    {
        MessageBox.Show("Failed to open RFID Reader connection");
    }
    else
    {
        ArrayList tagData = new ArrayList();
        if(!reader.ReadUserData(tagData))
        {
            MessageBox.Show("ReadUserData method failed");
        }
        txtReadUserData.Text = ByteArrayListToString(tagData);
        reader.Close();
    }
    // C# code specific to form and textbox operation
    this.Cursor = Cursors.Default;
}
}
```

# RFID Reader – Tag Read Calls

Intel\_CustomUI.frmMain

frmMain\_Load(object sender, System.EventArgs e)

```
private void btnReadTag_Click(object sender, System.EventArgs e)
{
    txtTagID.Text = wrapper.ReadTag(5000);
}

private void button1_Click(object sender, System.EventArgs e)
{
    txtUserData.Text = HextoAscii(wrapper.ReadUserData());
}

private void btnWriteData_Click(object sender, System.EventArgs e)
{
    wrapper.WriteUserData(txtWriteData.Text);
}

private string HextoAscii(string hex)
{
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i <= hex.Length - 2; i += 2)
    {
        sb.Append(Convert.ToString(Convert.ToChar(Int32.Parse(hex.Substring(i, 2), System.Globalization.NumberStyles.HexNumber))));
    }
    return sb.ToString();
}
```



# Intel Mobile Platform SDK Lab – RFID

## Software and Solutions Group

**Rajshree Chabukswar, Sr. Application Enabling Engineer**  
*SSG/GDRD/CET/Mobile Software Technology*

**Lester Memmott, Software Architect**  
*SSG/GDRD/CET/Mobile Software Technology*

**Highland Mary Mountain, Sr. SW Engineer/Architect**  
*SSG/GDRD/Digital Health Enabling*

*Month DD, 2006*

Please remember to turn in  
your session feedback form



# Executive Summary

- Full Lab Description (final wording, tbd):

Attendees will have the opportunity to have a hands-on experience using the Intel® Mobile Platform SDK to solve several of the problems that mobile application developers face today. Half of the lab will focus on the new RFID capabilities in which attendees will use RFID readers and RFID tags to write an application for a digital health use case. The other half of the lab will focus on using the Location APIs within the SDK. The attendee will create a Location-aware native application “mashing-up” several Web 2.0 technologies from Amazon, Yahoo! Local, and Google Maps. While the primary focus will be on new features in v1.2 such as Location and RFID, there will be opportunities to experiment with features from previous releases as well. Good preparation for this lab is the Intel® Mobile Platform SDK v1.2 presentation given earlier at the Software Enabling Summit.



# C# - IMPSDK Interface

```
Intel_CustomUI.frmMain frmMain_Load(object sender, System.EventArgs e)

private void frmMain_Load(object sender, System.EventArgs e)
{
    StatusBarPanel networkStatus = new StatusBarPanel();
    statusBar1.Panels.Add(networkStatus);

    StatusBarPanel batteryStatus = new StatusBarPanel();
    statusBar1.Panels.Add(batteryStatus);

    StatusBarPanel rfidStatus = new StatusBarPanel();
    statusBar1.Panels.Add(rfidStatus);

    rfidStatus.Text = wrapper.GetManufacturer() + " : " + wrapper.GetName();
    rfidStatus.ToolTipText = "BaudRate = " + wrapper.GetBaudRate().ToString() + "\nSerial Port = " + wrapper.GetSerialPort().ToString();

    ContextClass myClass = new ContextClass();
    ConnectivityInstance myInstance = (ConnectivityInstance) myClass.GetInstance("Connectivity");
    PowerInstance powInstance = (PowerInstance) myClass.GetInstance("Power");

    networkStatus.Text = myInstance.IsConnected.GetValue() ? "Connected" : "Disconnected";
    batteryStatus.Text = powInstance.PercentRemaining.GetValue().ToString() + "%";

    theObserver = new ConnectObserver();
    myInstance.Connected.AddObserver(theObserver);
    myInstance.Disconnected.AddObserver(theObserver);

    batteryObs = new BatteryObserver();
    powInstance.PercentRemaining.Changed.AddObserver(batteryObs);
    batteryObs.BatteryEvent += new BatteryObserver.BatteryEventHandler(BatteryEventHandler);

    theObserver.MyEvent += new ConnectObserver.MyEventHandler(EventHandler);
}
}
```