APPLICATION NOTE 7388

APPLYING THE DS2488 'C' DEMO

By: Stewart Merkel

Abstract:

A true wireless stereo (TWS) can easily integrate the DS2488 1-Wire Dual-Port Link IC in earbuds. This application note provides a demonstration, written in 'C,' of the basic building block routines.

Introduction

The headsets of today are increasingly using TWS Bluetooth earbuds to cut the cord of wires. However, contacts are needed for charging and communication between the earbuds and the charger box, and it is beneficial for these systems to have minimal connections. The DS2488 accomplishes this with just two contacts. This is a 'C' demonstration program that uses the DS2488EVKIT to emulate some of the interaction between a charger box and earbuds in a TWS headset application. These basic building block routines can be reused by software developers as a starting point.

Demonstration Setup

The following sections contain the information on demonstration setup.

C-Demo Files

Download the DS2488 'C' demo files

DS2488_Charge_Box files

'Release\DS2488_Charge_Box.exe' - compiled executable Charge Box program

'DS2488_Charge_Box.sln' - Visual Studio 2015 solution file for project

'DS2488_Charge_Box.vcxproj' - Visual Studio project file

'DS2488_Charge_Box_Demo.c' - Main charge box demo file

'DS2488.c' - DS2488 command functions module

'DS2488.h' - Header

'serial_win32ex.c' - Low Level functions to control UART.

'serial_win32ex.h' - Header

'1wire_UART.c' - Low level functions to use a UART peripheral as a 1-Wire® Master

'1wire_UART.h' - Header

DS2488_Earbud files

'Release\DS2488_Earbud.exe' - compiled executable Earbud program

'DS2488_Earbud.sln' - Visual Studio 2015 solution file for project

'DS2488_Earbud.vcxproj' - Visual Studio project file

'DS2488_Earbud.c' - Main earbud demo file

'DS2488.c' - DS2488 command functions module

'DS2488.h' - Header

'serial_win32ex.c' - Low Level functions to control UART.

'serial_win32ex.h' - Header

'1wire_UART.c' - Low level functions to use a UART peripheral as a 1-Wire Master

'1wire_UART.h' - Header

Setting Up the DS2488EVKIT

1. Set the DS9401 to 3.3V 1-Wire voltage by populating jumper JB1 (see **Figure 1**). The DS9401 emulates the charging box.

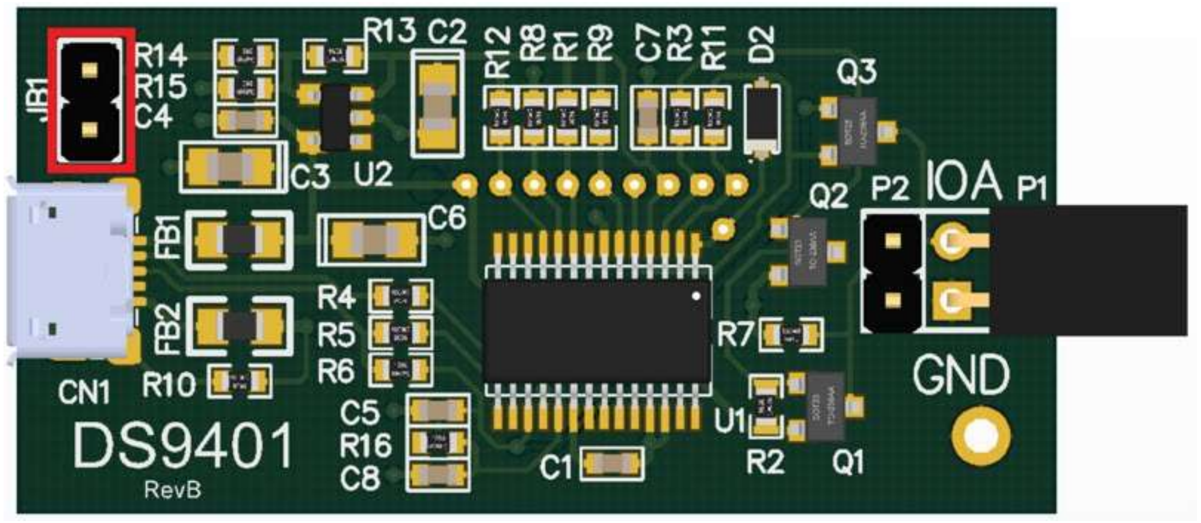


Figure 1. DS9401 board.

2. Set the DS2488EVKIT's PIOA' pin to ground on P2 by a jumper (pin 9–11) for right earbud detection. **Note**: The PIOA signal is already pulled high for the left earbud detection.
3. Populate DS2488EVKIT's JB1, 2, 3, 4, 5, and 6 and connect to DS9401 (i.e., emulating earbuds in the charging box). See **Figure 2**.

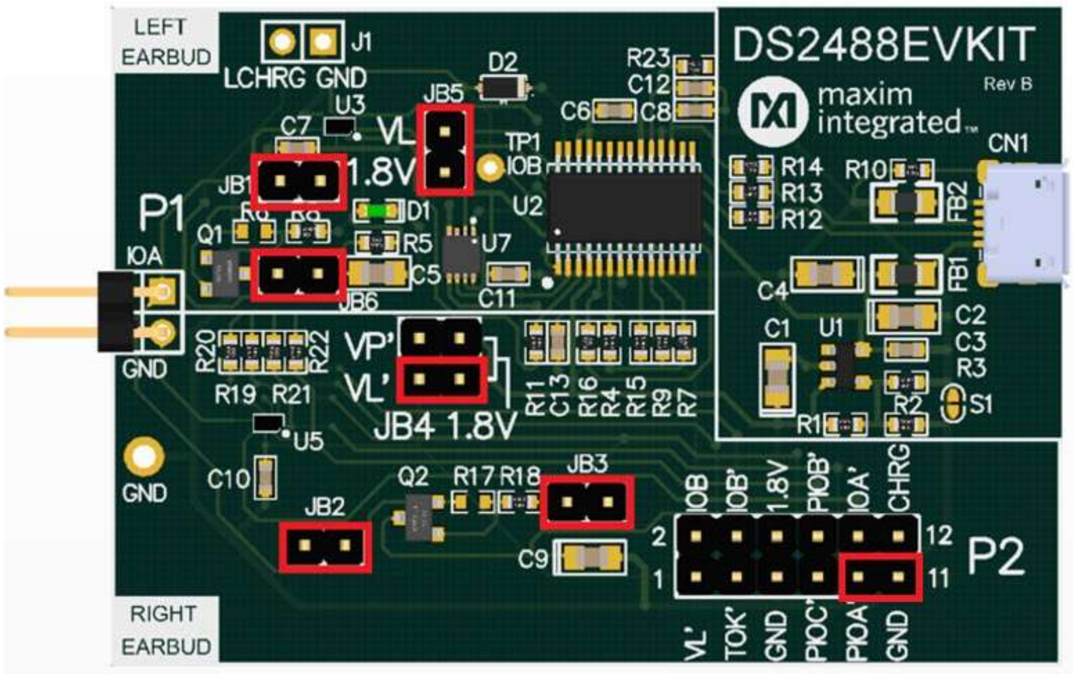


Figure 2. DS2488 EV kit board.

Note: Do not populate JB2 and JB3 to show true not in the charge box condition. The DS2488 Earbud program can then properly detect the earbuds out of charge box condition.

4. Install the Prolific drivers (if not already installed)
5. Connect the DS9401's P1 to DS2488EVKIT's P1 port.
6. Connect the DS9401 and DS2488EVKIT boards to the PC using the two USB cables.

Running the Demonstration

The 'C' demo can be run by executing the DS2488_Charge_Box.exe and DS2488_Earbud.exe in their own DOS command prompt application. The steps are as follows:

1. Open the Windows Device Manager (see **Figure 3**) and determine the associated COM port for each program (i.e., DS9401 and DS2488EVKIT).

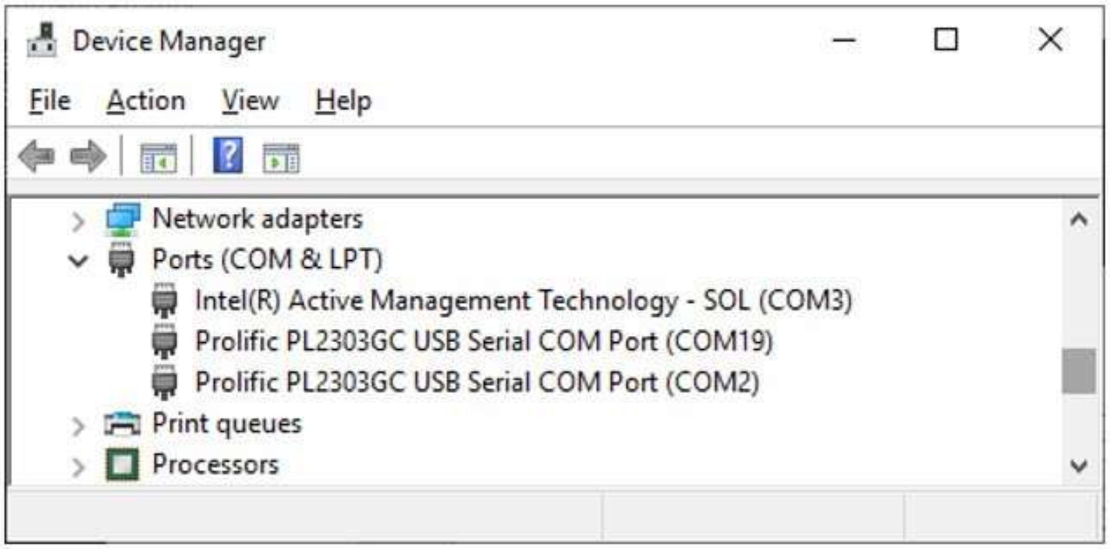
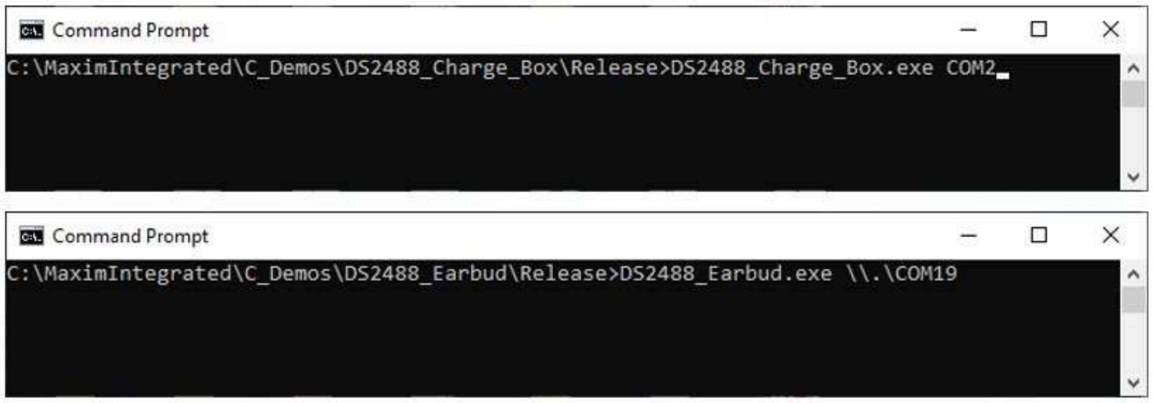

Device Manager
File Action View Help
Network adapters
Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
Prolific PL2303GC USB Serial COM Port (COM19)
Prolific PL2303GC USB Serial COM Port (COM2)
Print queues
Processors

Figure 3. Windows device manager.

2. Open the two DOS command prompts and navigate to each program's unzipped location. Then, execute the 'DS2488_Charge_Box.exe' and the 'DS2488_Earbud.exe' passing the COM port as a parameter as shown in **Figure 4**.

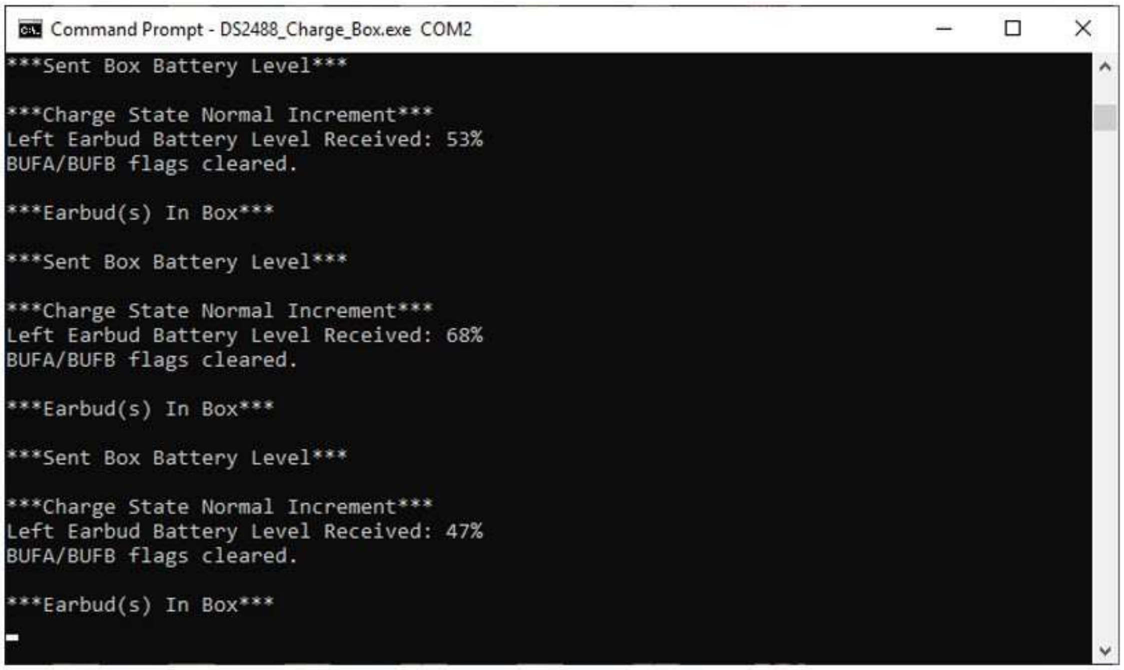


```
C:\MaximIntegrated\C_Demos\DS2488_Charge_Box\Release>DS2488_Charge_Box.exe COM2
```

```
C:\MaximIntegrated\C_Demos\DS2488_Earbud\Release>DS2488_Earbud.exe \\.\COM19
```

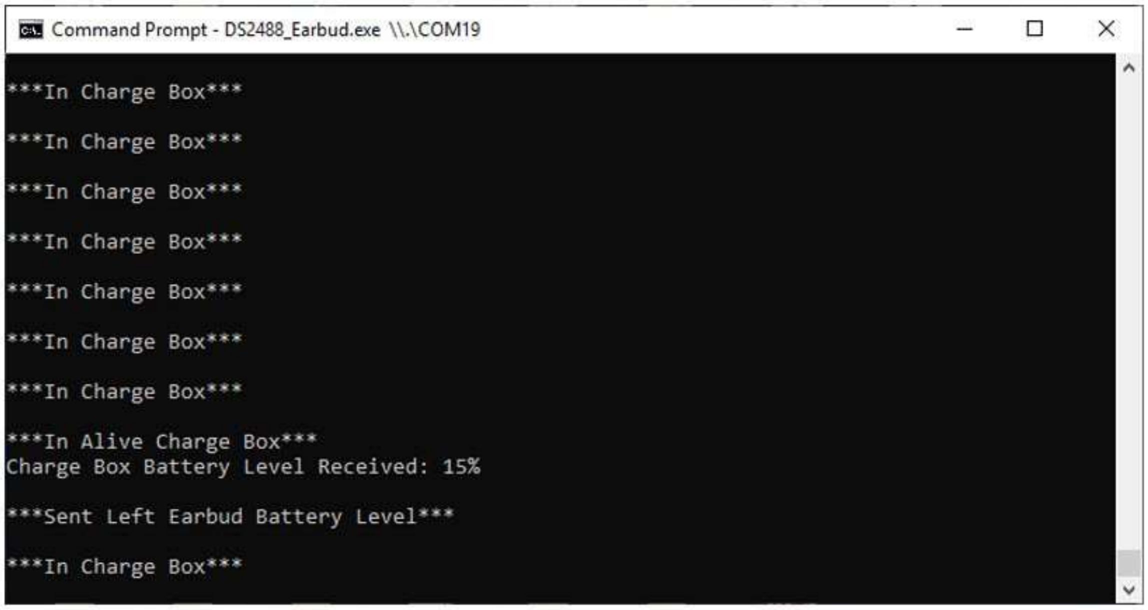
Figure 4. DOS command prompts.

3. Observe the constant messaging that occurs between the charge box and earbuds. Each program can be exited at any time by pressing the 'q' key.



Command Prompt - DS2488_Charge_Box.exe COM2

```
***Sent Box Battery Level***

***Charge State Normal Increment***
Left Earbud Battery Level Received: 53%
BUFA/BUFB flags cleared.

***Earbud(s) In Box***

***Sent Box Battery Level***

***Charge State Normal Increment***
Left Earbud Battery Level Received: 68%
BUFA/BUFB flags cleared.

***Earbud(s) In Box***

***Sent Box Battery Level***

***Charge State Normal Increment***
Left Earbud Battery Level Received: 47%
BUFA/BUFB flags cleared.

***Earbud(s) In Box***
```



Command Prompt - DS2488_Earbud.exe \\.\COM19

```
***In Charge Box***

***In Charge Box***

***In Charge Box***

***In Charge Box***

***In Charge Box***

***In Charge Box***

***In Charge Box***

***In Alive Charge Box***
Charge Box Battery Level Received: 15%

***Sent Left Earbud Battery Level***

***In Charge Box***
```

Note: Computer virtual COM parts are slow so this demo reacts slowly and message delays are set very long to accommodate ~11ms between bits.

Status Messages by the Demo

This section shows some of the status messages achieved by the demo between the earbuds and the charging box (or 'box'). The box or earbuds can send the charge state, battery information, and in/out box messages to one another. Following is a list of the continuous status messages that are communicated:

Box messages when earbuds in box with battery power available:

- Earbud(s) In Box
  - LEFT EARBUD: "64-bit ROM ID"
  - RIGHT EARBUD: "64-bit ROM ID"
- Sent Box Battery Level
- Charge State Normal Increment
  - LEFT Earbud Battery Level Received: XX%
  - BUFA/BUFB flags cleared

Earbud messages when in box with battery power available:

- In Charge Box
- Alive InCharge Box
  - Charge Box Battery Level Received: XX%
- Sent Left Earbud Battery Level

Box messages when earbuds have no battery power in box:

- Earbud(s) In Box
  - 1st Earbud Battery Dead
  - 2nd Earbud Battery Dead
- Charge State Long Increment

Earbud messages when in box with no battery power:

- Entered Left Earbud Loop
- Left Earbud ROMID
  - "64-bit ROM ID"
- In Charge Box (i.e., but with depleted battery as no alive messages received)

Box messages when earbuds not in box:

- Earbud(s) Not in Box - In Power Saver Mode

Earbud messages when not in Box:

- Not in Charge Box

Demo API

The main 'C' files of 'DS2488_Charge_Box.c' and 'DS2488_Earbud.c' utilize the API from 'DS2488.c,' '1wire_UART.c,' and 'serial_win32ex.c' modules to provide continuous status messages. The file functions in these API can be applied to other platforms to accelerate development of use cases using the DS2488 (e.g., TWS applications). The API source code provided has a license that is as close to being public domain as possible. Developers are free to use, change, and integrate this code into their applications without restrictions. Following is an overview description of the API available:

DS2488.c

This is the core API to control and use the DS2488 device.

WriteConfiguration - Performs writing to configuration register.
ReadConfiguration - Performs reading from configuration register.
WriteBuffer - Performs writing to the 8-byte buffer.
ReadBuffer - Performs reading from the 8-byte buffer.
ReadStatus - Performs reading out data from the status register.
PIO_Write - Sets the PIO pin output values.
PIO_Read - Provides PIO pin input values.
WriteTimeoutValue - Sets timer duration = TVAL x 100us.
ReadTimeoutValue - Reads timer duration = TVAL x 100us.
setDeviceSelectMode - Selects the 1-Wire ROM command for all Device Function commands.
DeviceSelect – Executes the selected ROM command.

1wire_UART.c

This is the fundamental API for 1-Wire communication by a UART. This API was developed per **Tutorials 214 (Using a UART to Implement a 1-Wire Bus Master)**.

OWReset - Resets all the devices on the 1-Wire Net and return the result.
OWTouchBit - Sends 1 bit to the 1-Wire Net and return the result 1 bit read from the 1-Wire Net.
OWTouchByte - Sends 8 bits to the 1-Wire Net and return the 8 bits read from the 1-Wire Net.
OWWriteByte - Sends 8 bits of communication to the 1-Wire Net and verify.
OWWriteBit - Sends 1 bit of communication to the 1-Wire Net.
OWReadBit - Sends and Return the result 1 bit read from the 1-Wire Net.
OWReadByte - Sends 8 bits of read to the 1-Wire Net and return the 8 bits read result.
OWBlock - Transfers a block of data to and from the 1-Wire Net.
OWSearch - Does a general search, F0h. This function continues from the previous search state.
OWFirst - Finds the 'first' device ROM number on the 1-Wire Net.
OWNext - Finds the 'next' devices ROM numbers on the 1-Wire Net.
OWVerify - Verifies the device with the ROM number in ROM_NO buffer is present.
OWTargetSetup - Setup search to find the device type 'family_code' on the next call to OWNext.
OWFamilySkipSetup - Setup search to skip the current device type on the next call to OWNext.

OWReadROM - Performs the read-ROM function 33h to read a ROM number and verify CRC8.
OWSkipROM - Performs a Skip ROM function CCh to select the all 1-Wire slaves at once.
OWMatchROM - Does a Match ROM 55h using the global ROM_NO device.
OWOverdriveSkipROM - Does an Overdrive Skip ROM 3Ch to set all 1-Wire slaves to overdrive.
OWResume - The function does a Resume command A5h.
OWOverdriveMatchROM - Does an overdrive Match ROM using the global ROM_NO device.

OWSpeed - Sets the 1-Wire Net communication speed (i.e., Standard or Overdrive).

Serial_win32ex.c

This provides the serial port communication API primitives for Windows 32-bit, which is a UART (e.g., Prolific PL2303GC).

FlushCOM - Flush the RX and TX buffers.

WriteCOM - Writes an array of bytes to the COM port, verifies that it was sent out.
ReadCOM - Reads an array of bytes to the COM port, verifies that it was sent out.
BreakCOM - Sends a break on the COM port for at least 2ms.
SetBaudCOM - Sets the baud rate on the COM port.
OpenCOM - Attempts to open a COM port. Keep the handle in ComID. Set the starting baud rate to 9600. Enable DTR and disable RTS.
CloseCOM - Closes the connection to the port.
DTRCOM - Sets the active low DTR state.
RTSCOM - Sets the active low RTS state.
DCDCOM - Reads the DCD status.
RICOM - Reads the active low RI status.
CTSCOM - Reads the active low CTS status.
DSRCOM - Reads the active low DSR status.

Conclusion

The TWS headset developers can start with the building blocks in the DS2488 'C' Demo program and begin writing feature code with the DS2488 'C' API. It also details several of the APIs so a software engineer knows what is available. Developing the 1-Wire communication code between a charging box and earbuds can easily be done with the provided API.

Trademarks

Related Parts	
DS2488	1-Wire Dual-Port Link