White Paper

Kanapathy Murugayah
Platform Solution Architect
Intel Corporation

Lee Zhan Qiang
Platform Application Engineer
Intel Corporation

The Intel® Quark SoC: Revolutionizing the Entry-Level Cash Register Market

July 2014

330974-001

Executive Summary

The low-cost entry-level segment of the retail sales cash register and cashiering devices market has always been dominated by products that are simultaneously low-cost, reliable, and highly secure.

The Intel® Atom Processor has generally been considered too expensive, large, and unnecessarily fast for these low-cost devices that are geared towards offline/non-connected, rudimentary sales transactions.

This use-case provided in this white paper details how the Intel® Quark System on a Chip (SoC) platform will galvanize the creation of a game-changing entry-level retail sales solution that is quick, full-featured, highly secure, *and* low-cost.

This is feasible for the first time due to the Quark's great-leap-forward combination of: Abundance in I/O interface, *and*, available peripheral options and software ingredients from embedded operating systems like Wind River* Intelligent Device Platform (IDP) 2.x*, Yocto*, Linux*, and Windows Embedded 8*.

The Intel® Embedded Design Center provides qualified developers with Web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://intel.com/embedded/edc.

Contents

Figures

Tables

Background

The entry-level retail sales cash registers market is no longer receptive to larger, powerful, more expensive machines. This is due to the competitive factors of initial cost, peripheral connectivity, and fiscal compliance. These factors inform the fundamental business reality of this market, both in emerging capital markets, and in established, 'first-world' countries.

The Intel Atom® Processor, Intel's heretofore most applicable offering tailored to this market, possesses all of the components required (in terms of hardware and software capability) to compete in this domain, particularly within the electronic cash register market. However, it has been hobbled by one main weakness: It is too expensive to compete in this demanding, "low-cost margin" arena of competition.

Solution

With the advent of the Quark SoC from Intel, the game has changed.

The Quark SoC enables a feature rich user interface by leveraging an ideal combination of strengths:

- **Lower Cost** (*Competes at the Low-Cost Price Point Businesses Demand*)
- **Simple, Versatile Configuration** (*Flexible, Open-Source support of existing Peripherals*)
- **Exceptional Performance** (*Cutting-Edge, newest Intel Processor Technology*)

This is due to the superior internal capabilities that are delivered by I/O wrapping within the Quark SoC.

This landmark Intel SoC creation foretells a Quark-centric flexible solution era within this product market.

It will mark a defining pivot point for retail sales Original Design Manufacturers (ODMs) and Original Equipment Manufacturers (OEMs).

Software Stack and Operating System

Quite arguably, software is at the very heartbeat of the most compelling and enticing computing and media devices.

Software powers the iconic products that eventually end up capturing the imagination of the general public, world-wide.

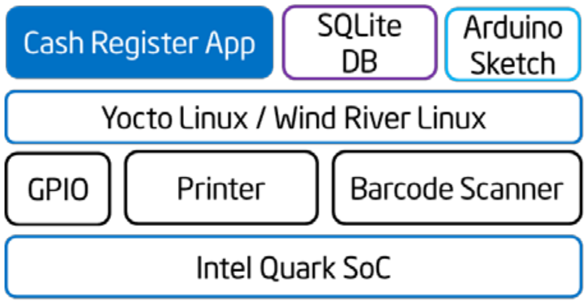
This is very likely because, without software, all that is seen and perceived by the consumer or business owner is a piece of inert, lifeless, boring hardware.

Thus, for the Quark SoC solution to be not only viable, but evolutionary, the Quark's Software Stack had to be assembled in a way that could showcase for the retail user an interface and feature experience that is dynamic, intuitive, logical, and speedy.

The Software Stack detailed in this Section does just that.

Figure 1 shows the Quark Software Stack. It uses Intel's Galileo Development Kit.

Figure 1. Cash Register Software Stack on the Intel Galileo Development Kit



This software stack exemplifies the groundbreaking software capabilities that were ushered in with the introduction of the Intel Galileo Development Kit (Intel® Quark^{TM} SoC X1000).

The cash register application is typical of those for cash registers.

It can be written using either C++* or Python*.

It reads the barcodes of products from barcode scanners and then makes queries to the lightweight SQLite Database that comes with Yocto Linux or with Wind River Linux IDP*. Every time a product is scanned, an internal socket sends the product name and product price to the Arduino Sketch Application. This drives the information to the Liquid Crystal Display (LCD). Once checkout is complete, the transaction receipt is printed via the thermal printer.

For this sequence just described, the following software codes (#1 and #2) are required (they can be accompanied by other, additional codes):

1. **Arduino* Socket LCD Sketch Program*** – The Arduino Socket LCD Sketch Program*, powered by The Arduino LiquidCrystal Library*, provides easy interfacing to I/O ports and sensors on the Intel Galileo Development Kit. The program behaves as a Transmission Control Protocol (TCP) Server, and waits for client to connect before driving the Liquid Crystal Display.

2. **C++/Python Cash Register Application*** – Establishes and creates connection with the locally running TCP server. Connects to the barcode scanner and thermal printer to enable reading the product barcodes. Queues and then prints the checkout transaction receipt.

3. **Fiscal Software Application** (*Optional*) – For countries that require fiscal compliance and other similar regulatory requirements.

Figure 2 shows sample code for the required Arduino Socket LCD Sketch Program.

Figure 2. Arduino Socket LCD Sketch Program: Sample Code

```
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Ethernet.h>
// initialize the library with the numbers of the interface pins for
LCD

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
// Setup MAC address and TCP Port
byte mac[] = {  0x98, 0x4F, 0xEE, 0x00, 0x86, 0x77 };
int serverPort=5555;
EthernetServer server(serverPort);

void setup() {

  Serial.begin(9600);
  lcd.init(1,12,255,11,5,4,3,2,0,0,0,0);
  lcd.begin(16, 4); // set up the LCD's number of columns and rows:
  lcd.setCursor(0,0);
  lcd.write("Welcome To"); //Simple welcome message on ECR power ON
  lcd.setCursor(0,1);
  lcd.write("Quark ECR");
  Serial.println("\nStarting Server!");
  server.begin(); //Start TCP Server
  Serial.println("Server started");

  }

void loop() {
  EthernetClient client = server.available();
  if (client) {
    String clientMsg ="";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        //Serial.print(c);
        clientMsg+=c;
        if (c == '\n') {
          Serial.println(clientMsg);//print it to the serial
          lcd.clear();
          lcd.setCursor(0,0);
          for (int i=0;i<clientMsg.length()-1;i++)
          {
          lcd.print(clientMsg[i]);
          }
          clientMsg="";
        }
      }
    }
    client.stop();
  }

}
```

The Galileo Board Support Package (BSP), Cash Register Application, and Customized Drivers for the Intel Galileo

The solution isn't entirely complete with the Arduino Sketch program alone.

The cash register application, additional drivers, and libraries, will require the Intel® Quark™ Board Support Package (BSP) and Build Guide. It is available here:

https://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=23197&keyword=quark&lang=eng

The Yocto Linux BSP* allows a developer to add additional embedded application, drivers, and load them into one very small bootable Linux* image for the Intel Galileo.

A free guide that details this, the Yocto Official Application Developer Guide*, can be found here:

http://www.yoctoproject.org/docs/1.6/adt-manual/adt-manual.html

To develop and install additional drivers (for instance: to add a Universal Serial Bus (USB) Printer driver), developers can refer to the blog article titled: *Intel Galileo – Building Linux* Image*. It located on the blog Sergey's Blog (Sergey Malinov, author), here:

http://www.malinov.com/Home/sergey-s-blog/intelgalileo-buildinglinuximage

Enabling the USB Thermal Printer

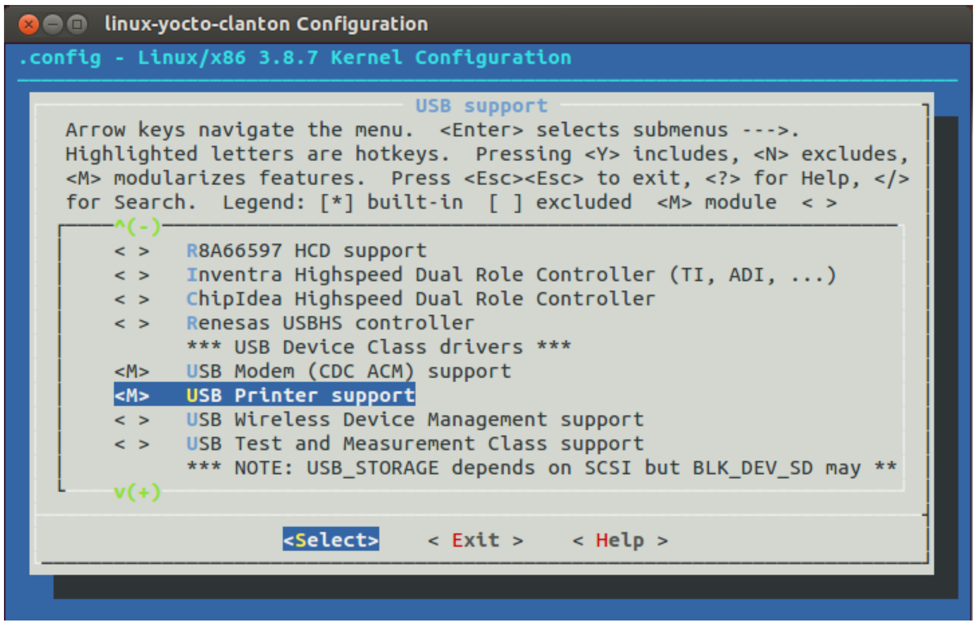
As described in Sergey Malinov's blog article, the only thing required to enable the USB Thermal Printer is to run the bitbake command: "`bitbake linux-yocto-clanton -c menuconfig`," within the terminal environment for the BSP being built.

1. Run the command: `bitbake linux-yocto-clanton -c menuconfig`

   Once run, a console appears that allows one to customize the drivers required for particular embedded operations.

2. Enable the USB printing capability by selecting **USB Printer support**, as shown in Figure 3.

Figure 3. Customizing the Yocto Linux Kernel* for Intel Galileo



```
linux-yocto-clanton Configuration
.config - Linux/x86 3.8.7 Kernel Configuration
                              USB support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
      ^(-)
      < >    R8A66597 HCD support
      < >    Inventra Highspeed Dual Role Controller (TI, ADI, ...)
      < >    ChipIdea Highspeed Dual Role Controller
      < >    Renesas USBHS controller
             *** USB Device Class drivers ***
      <M>    USB Modem (CDC ACM) support
      <M>    USB Printer support
      < >    USB Wireless Device Management support
      < >    USB Test and Measurement Class support
             *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may **
      v(+)

                  <Select>    < Exit >    < Help >
```

Enabling USB Barcode Scanner

The USB barcode scanner works like a Human Interface Device (HID) keyboard input system.

The HID drivers are already included as part of the default Yocto* Linux Build* for Intel Galileo.

To enable the scanner to read a barcode, an additional Python* module may be required. This module translates every chunk of data into readable text to enable the data to be used to query the SQLite Database.

Sample Python* program code is provided in Figure 4.

This code demonstrates how to enable barcode scanning, using the Linux* Event Interface*.

This sample code is from the Python-Evdev Library*. This library is not included in the Intel® Quark™ BSP Distribution Kit.

The recipe script can be downloaded here:
https://github.com/wallacezq/recipes-python-evdev.

Refer to the readme file for Installation guidance.

Alternatively, for users who wish to do the same thing in Arduino Sketch*, an Arduino Barcode Library* is available here:
https://github.com/wallacezq/HIDBarcodeScanner.

Figure 4. Sample Python Code for the Barcode Scanner

```
#!/usr/bin/env python
from evdev import InputDevice, ecodes, categorize
import signal, sys

scancodes = {
    # Scancode: ASCIICode
    0: None, 1: u'ESC', 2: u'1', 3: u'2', 4: u'3', 5: u'4', 6: u'5', 7:
u'6', 8: u'7', 9: u'8',
    10: u'9', 11: u'0', 12: u'-', 13: u'=', 14: u'BKSP', 15: u'TAB', 16:
u'Q', 17: u'W', 18: u'E', 19: u'R',
    20: u'T', 21: u'Y', 22: u'U', 23: u'I', 24: u'O', 25: u'P', 26: u'[',
27: u']', 28: u'CRLF', 29: u'LCTRL',
    30: u'A', 31: u'S', 32: u'D', 33: u'F', 34: u'G', 35: u'H', 36: u'J',
37: u'K', 38: u'L', 39: u';',
    40: u'"', 41: u'`', 42: u'LSHFT', 43: u'\\', 44: u'Z', 45: u'X', 46:
u'C', 47: u'V', 48: u'B', 49: u'N',
    50: u'M', 51: u',', 52: u'.', 53: u'/', 54: u'RSHFT', 56: u'LALT', 100:
u'RALT'
}

dev = InputDevice('/dev/input/event3')

def signal_handler(signal, frame):
    print 'Stopping'
    dev.ungrab()
    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

dev.grab()

barcode = ""
for event in dev.read_loop():
    if event.type == ecodes.EV_KEY:
        data = categorize(event)
        if data.keystate == 1 and data.scancode != 42: # Catch only keydown,
and not Enter
            if data.scancode == 28:
                print barcode
                barcode = ""
            else:
                barcode += scancodes[data.scancode]
```

Hardware Setup and I/O Availability

This section describes how to set up the Hardware and I/O Availability.

Getting around within the Intel® Galileo Development Board Kit

The Intel® Galileo Development Board (the Galileo is based on the Intel® Quark^{TM} SoC X1000, a 32-bit Intel® Pentium® processor-class system on a chip). It is the first Intel® architecture-based board that is designed to be hardware and software pin-compatible with shields for the Arduino Uno* R3*.

The board comes standard with the following compatibility:

- Full-size mini-PCI Express Slot
- 100 Mb Ethernet Port
- Micro-SD Slot
- RS-232 Serial Port
- USB Host Port
- USB Client Port, and,
- 8 MB NOR Flash*

To develop a detailed understanding of the I/Os and interfaces that come standard with the Galileo Board, and associated hardware connectivity, refer to this document:
http://arduino.cc/en/ArduinoCertified/IntelGalileo

Table 1 provides a reference summary of how the I/Os are used for hardware connectivity.

Table 1. I/Os Supported by the Galileo Development Kit

I/O	Pin outs	Function(s)
Digital Port	Digital 0 - 13	Configured to drive the LCD
Analog Port	A0 to A5	Configurable to accept analogue signal referencing to either 3.3 V or 5 V and I^2C signal interface (A4 and A5)
USB Port	Client Port and Host Port	Client Port : Arduino IDE interface for programming I/Os Host Port: Interfacing USB Peripherals (external storage device, barcode scanner, printer, Near Field Communication (NFC) reader/writer, and so on).
LAN Port		Connectivity to Cloud or Backend Database

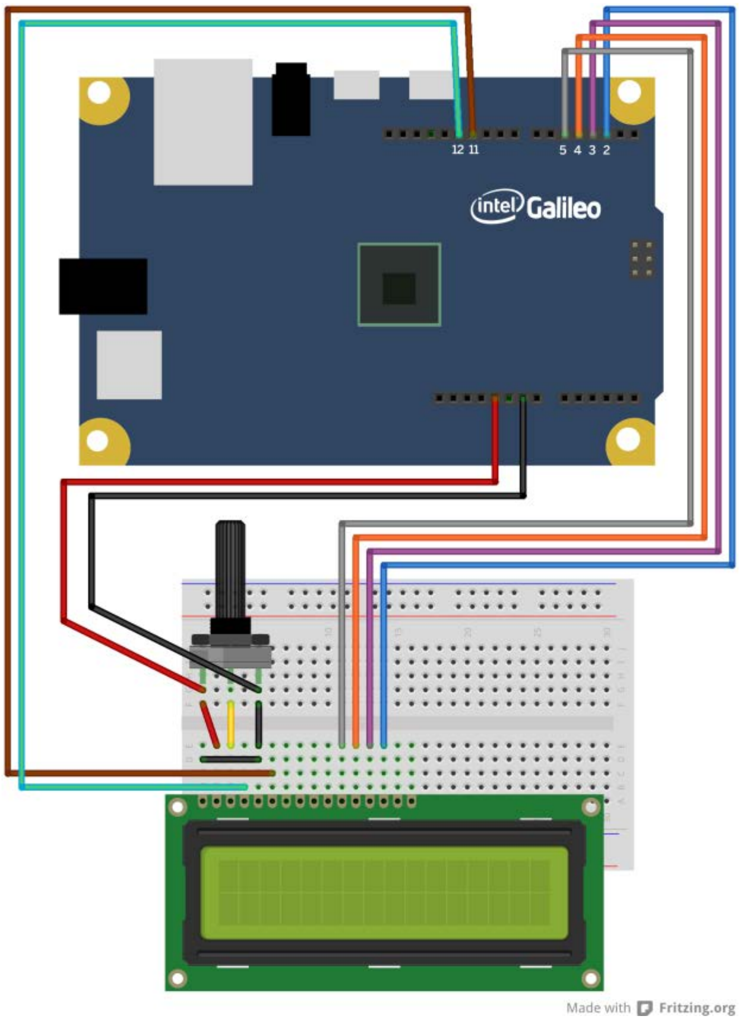
Interfacing Liquid Crystal Display with Digital I/Os

Interfacing an LCD display with the Intel Galileo is a simple task.

Figure 5 illustrates this. One simply connects the typical LCD (in this illustration, the Hitachi* HD44780*).

For more extensive information, and to thoroughly familiarize oneself with the basics of LCD utilizing the Arduino IDE*, refer to the following tutorial: (http://arduino.cc/en/Tutorial/LiquidCrystal)

Figure 5. LCD Display Interfaced with the Intel Galileo



Interfacing Barcode Scanner and Printers

External retail peripheral interfacing can be easily accomplished via the USB port, as an Operating System (OS) is available.

The only thing needed is the driver software and the application software to claim those devices based on the nature of each peripheral's function.

The Barcode Scanner and the external Thermal Printer use the USB port (host-port). In the case of a limited number USB ports, a USB hub can be used to accommodate both devices.

USB hub software drivers are included in the default Yocto Board Support Package (BSP)* for Intel Galileo.

§

Conclusion

This white paper details how the Quark SoC platform delivers an ultra-competitive combination of diverse hardware I/Os and connectivity capability for connected entry-level cash registers.

Using applicable software elements, the Quark SoC solution for these entry-level devices can be customized based on the product requirements and sales scenarios.

Leveraging the tiny, customizable operating system that can be developed with the Yocto BSP* for Galileo, ODMs and System Integrators (SIs) can now simultaneously: Add driver support for sought after retail sales peripherals, embed a lightweight SQLite database, and incrementally scale their provided solution.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://www.intel.com/p/en_US/embedded

Authors

Kanapathy Murugayah is a Platform Solution Architect at Intel Corporation.

Lee Zhan Qiang is a Platform Application Engineer at Intel Corporation.

Acronyms

Table 2. Acronyms

Acronym	Definition
POS	Point of Sale
SoC	System on a Chip
LAN	Local Area Network
LCD	Liquid Crystal Display
GPIO	General Purpose Input Output
USB	Universal Serial Bus
BSP	Board Support Package
ODM	Original Design Manufacturer
OEM	Original Equipment Manufacturer
TCP	Transmission Control Protocol
IDP	Intelligence Device Platform
HID	Human Interface Device
NFC	Near Field Communication
OS	Operating System
SI	System Integrator

References

1. Intel Galileo Development Kit:
   http://www.intel.com/content/www/us/en/do-it-yourself/galileo-maker-quark-board.html
2. Intel Galileo Schematics:
   https://communities.intel.com/community/makers/documentation/galileodocuments
3. Intel Galileo (Arduino site):
   http://arduino.cc/en/ArduinoCertified/IntelGalileo
4. Yocto Project : www.yoctoproject.org
5. Wind River Linux IDP 2.0: www.windriver.com/idp/
6. Arduino LCD Tutorial : http://arduino.cc/en/Tutorial/LiquidCrystal
7. Intel Galileo LCD Interface Guide:
   http://makezine.com/2014/02/03/galileo-project-how-many-days-until-make-comes-out/
8. Python Programming Language: www.python.org
9. Galileo Linux Image Customization:
   http://www.malinov.com/Home/sergey-s-blog/intelgalileo-buildinglinuximage