HOLT INC. INTEGRATED CIRCUITS

August 12, 2011

AN - 166

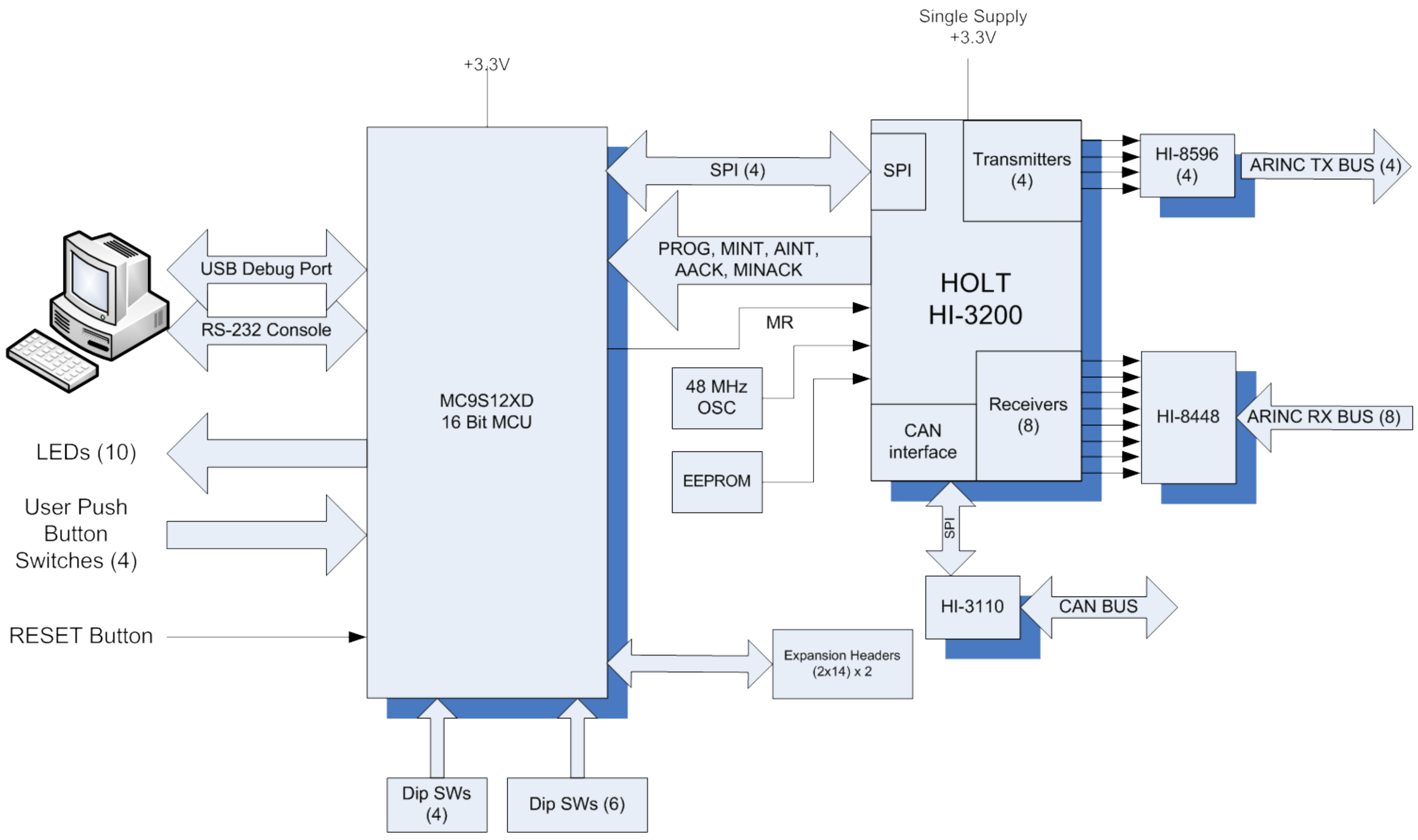
HI-3200 Avionics Data Management Engine Evaluation Board – Software Guide

Introduction

This application note provides more detail on the HI-3200 demo software provided in the Holt HI-3200 ARINC 429 Evaluation Kit.

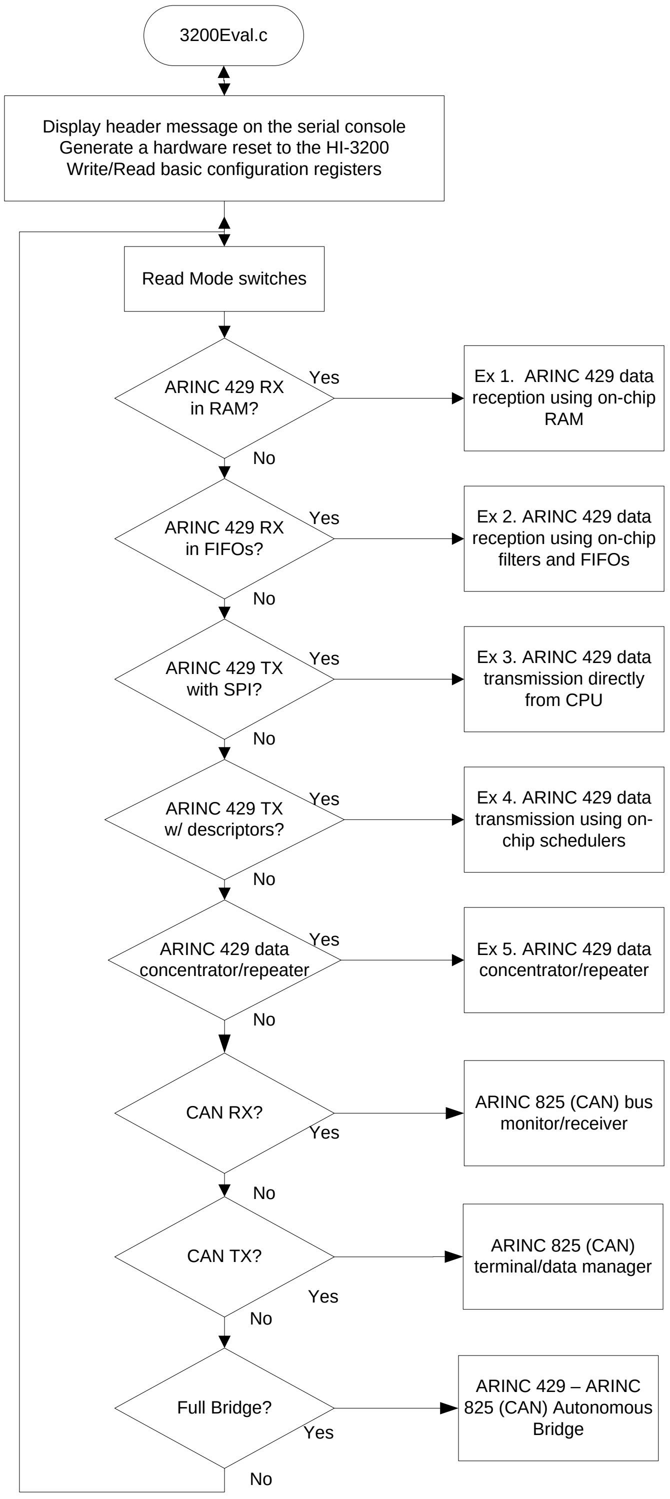
A Quick Start Guide and a User's Guide for HI-3200 evaluation kit can be found on the CD-ROM. Use these guides to become more familiar with the board setup and operation of the demo software.

Evaluation Board Block Diagram



Demo software overview

This overview flow chart shows the demo program at a glance.


3200Eval.c
Display header message on the serial console
Generate a hardware reset to the HI-3200
Write/Read basic configuration registers
Read Mode switches
ARINC 429 RX in RAM?
Yes
No
Ex 1. ARINC 429 data reception using on-chip RAM
ARINC 429 RX in FIFOs?
Ex 2. ARINC 429 data reception using on-chip filters and FIFOs
ARINC 429 TX with SPI?
Ex 3. ARINC 429 data transmission directly from CPU
ARINC 429 TX w/ descriptors?
Ex 4. ARINC 429 data transmission using on-chip schedulers
ARINC 429 data concentrator/repeater
Ex 5. ARINC 429 data concentrator/repeater
CAN RX?
ARINC 825 (CAN) bus monitor/receiver
CAN TX?
ARINC 825 (CAN) terminal/data manager
Full Bridge?
ARINC 429 – ARINC 825 (CAN) Autonomous Bridge

The program enters the desired mode selected by the mode switches. To restart into a different mode, reconfigure the mode switches and reset the board.

MCU Clock and SPI Frequencies

The Freescale MC9S12XDT512 (MCU) on the main board uses a 4MHz crystal for operation and the built-in PLL multiplies this by 20 to achieve an 80MHz system clock. This system clock is divided by two for a 40MHz Bus Clock which is used internally for the MCU peripherals.

The PLL is programmed to multiply by 20 by this line of code in the 3200Eval.c, Peripherals module:

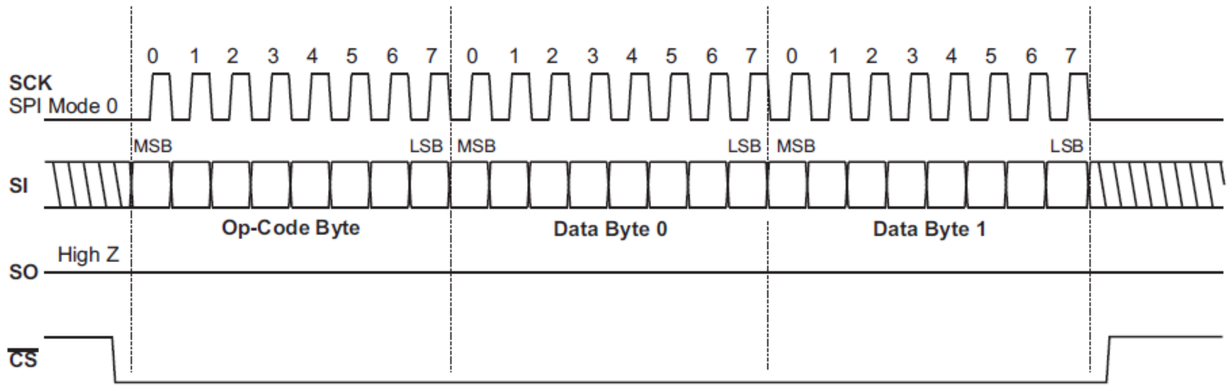
```
        SYNR = 9;                  // 80Mhz PLL system clock
```

The maximum SPI frequency for the HI-3200 is 20MHz.

Special handling of the /SS SPI signal:

All HI-3200 SPI Op-Codes require the /CS to remain low for the complete duration of the data transfer including multi-byte reads and writes. Refer to figures 6 and 7 of the data sheet for timing diagram examples.

To achieve this, the default SPI slave select line /SS in the Freescale MCU must be reconfigured as a GPIO and controlled by code in the function. This technique is common for devices requiring the /CS line to remain low during multi-byte transfers. The first positive SCK edge must occur after /CS is asserted low; the last falling SCK edge must occur before the /CS is negated high as shown in the following diagram:



There are functions that read a single byte from the HI-3200 SPI port, write a command to the HI-3200 SPI port and a few others which read or write a command plus a multiple number of bytes.

For example the function below is the basic function to write a command plus two bytes of data to the HI-3200 SPI port.

```
void writeMAP (unsigned short addrMAP) {

  unsigned char dummy;

  // disable auto /SS output, reset /SS Output Enable
  SPI0CR1 = SPI0CR1 & ~SPI0CR1_SSOE_MASK;
  // disable auto /SS output, reset SPI0 Mode Fault
  SPI0CR2 = SPI0CR2 & ~SPI0CR2_MODFEN_MASK;
  // assert the SPI0 /SS strobe
  SPI0_nSS = 0;
  // send op code (ignore returned data byte)
  dummy = txrx8bits_8(0x8C,1);

    // send next byte (ignore returned data byte)
    dummy = txrx8bits_8((char)((addrMAP>>8) & 0xFF),1);
    // send LS byte (ignore returned data byte)
    dummy = txrx8bits_8((char)(addrMAP & 0xFF),1);


  // de-assert the SPI0 /SS strobe
  SPI0_nSS = 1;
  // enable auto /SS output, set /SS Output Enable
  SPI0CR1 = SPI0CR1 | SPI0CR1_SSOE_MASK;
  // enable auto /SS output, set SPI0 Mode Fault
  SPI0CR2 = SPI0CR2 | SPI0CR2_MODFEN_MASK;

}
```

The function below is the basic function to write a command and read two bytes of data on the HI-3200 SPI port.

```
unsigned long readMAP (void) {

  unsigned short j, rxdata;  // long = 32 bits

  // disable auto /SS output, reset /SS Output Enable
  SPI0CR1 = SPI0CR1 & ~SPI0CR1_SSOE_MASK;
  // disable auto /SS output, reset SPI0 Mode Fault
  SPI0CR2 = SPI0CR2 & ~SPI0CR2_MODFEN_MASK;
  // assert the SPI0 /SS strobe
  SPI0_nSS = 0;

  // send op code (ignore returned data byte)
  rxdata = txrx8bits_8(0x88,1);
  // send dummy data / receive, left-shift
  j = txrx8bits_8(0x00,1);
  rxdata = (j << 8);
  // send dummy data / receive and OR the least signif. byte
  j = txrx8bits_8(0x00,1);
  rxdata = rxdata | j;

  // de-assert the SPI0 /SS strobe
  SPI0_nSS = 1;
  // enable auto /SS output, set /SS Output Enable
  SPI0CR1 = SPI0CR1 | SPI0CR1_SSOE_MASK;
  // enable auto /SS output, set SPI0 Mode Fault
  SPI0CR2 = SPI0CR2 | SPI0CR2_MODFEN_MASK;
  return rxdata;

}
```

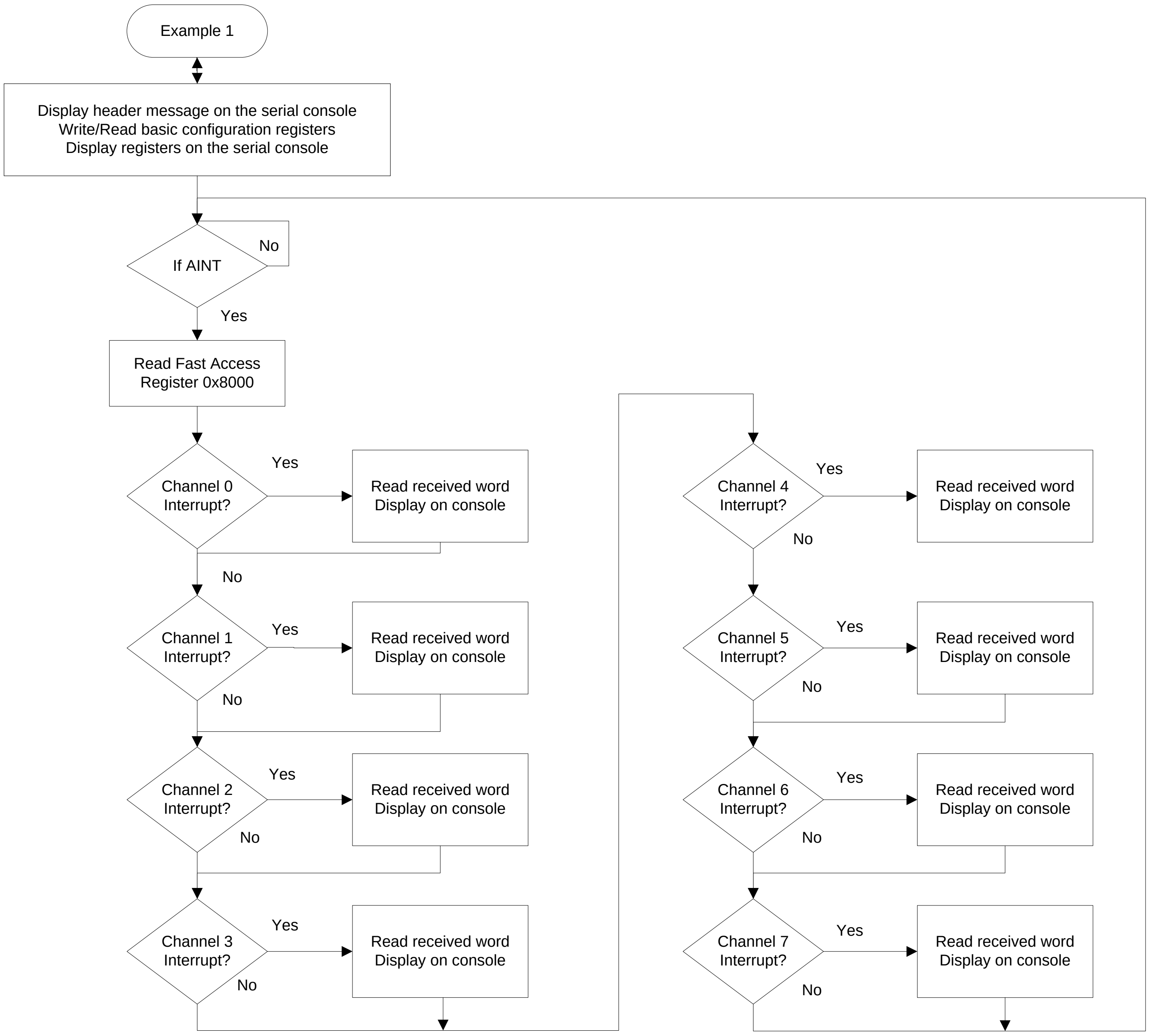
See the HI3200Eval.h header file for the options available in the define statements.

Uart.c Serial Port (RS-232)

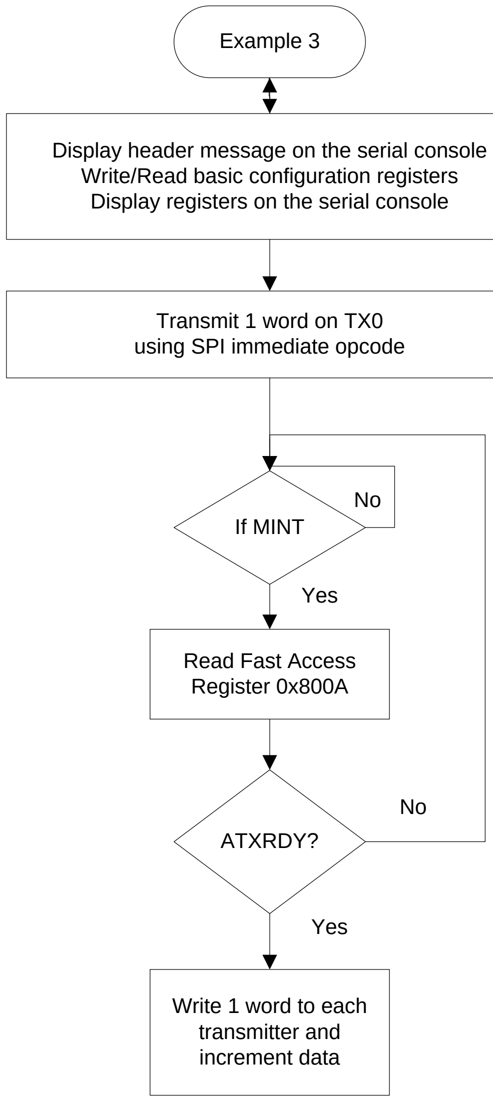
The drivers to support the serial port (Console) are contained in this module. There are some function drivers to allow messages to be sent and received on the UART. This is useful to log status or data messages on HyperTerminal or any other terminal program. It currently uses polling to determine when the data receive or transmit registers can be read or written.

The following two flow-charts illustrate the program flow for those two main demo modes.

Flow-chart of Mode - 0 (Ex 1. ARINC 429 Receive in RAM)


Example 1
Display header message on the serial console
Write/Read basic configuration registers
Display registers on the serial console
If AINT
No
Yes
Read Fast Access Register 0x8000
Channel 0 Interrupt?
Channel 1 Interrupt?
Channel 2 Interrupt?
Channel 3 Interrupt?
Channel 4 Interrupt?
Channel 5 Interrupt?
Channel 6 Interrupt?
Channel 7 Interrupt?
Read received word Display on console

Flow-chart of Mode - 4 (Ex 3. ARINC 429 Transmit with SPI)


Example 3
Display header message on the serial console
Write/Read basic configuration registers
Display registers on the serial console
Transmit 1 word on TX0
using SPI immediate opcode
If MINT
No
Yes
Read Fast Access
Register 0x800A
ATXRDY?
No
Yes
Write 1 word to each
transmitter and
increment data

HI-3200 demo Codewarrior Software Project

The software project is built with Freescale's CodeWarrior version 5.9.0 using the free limited 32K version. The main functions are in 3200eval.c. The software project "HI-3200 Demo" will normally be distributed in a zip file on a CD-ROM with the same name. **To develop, debug and download this software into the board a PE Micro "USB Multilink Interface" debug cable is necessary. It is not provided in this kit.** To purchase this cable, go to the PE Micro website or purchase it from Digi-Key. See the links at the end of this document.

Project Files

Source Files

3200eval.C	Main code
Uart.c	Low-level UART drivers
datapage.c	Freescale™ IDE support file

Include Files

HI3200eval.h	HI-3200 header
Uart.h	
Common.h	Common defines for the project
Derivative.h	Freescale™ IDE support file
Mc9s12xdt512.h	Freescale™ IDE target part support file

CodeWarrior and Software Project Setup:

1. Download and install the CodeWarrior IDE from the Freescale website. The download links are provided below.
2. Unzip the HI-3200 zip file into the directory you plan to use for your project.
3. Navigate to the HI-3200 project folder and double click the HI-3200 Demo.mcp project file to launch this project with CodeWarrior. The IDE should open with the project files on the left side of the window.
4. Click Make from the Project menu to rebuild the project. The project should build without errors. You may receive a dead assignment warning if for example some defines are set to a zero value.
5. Install the PE Micro USB Multilink Interface cable per the instructions.
6. Plug the USB Multilink 6-pin debug cable into the J9 debug connector and power up the board with 3.3V.
7. Download the program by clicking Debug from the Project menu. The first time you download you may need to configure the debugger for the USB Multilink cable. After downloading is complete the debugger window should be displayed with the first line in main.c highlighted. Press the green arrow button to run the program. Since the program has been loaded you can power down the board and re power the board and the program should run automatically without the debugger.

Freescale MC9S12XDT512xxx Development Tools

The Freescale microcontroller data sheet and other documentation can be found at this link:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=S12XD&tid=16bhp

If these links become out of date go to: http://www.freescale.com/

and search for information on "S12XD: 16-Bit Automotive Microcontroller".

A Free 32K limited version of the Code Warrior IDE from Freescale is available:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-HCS12X&fsrch=1

The US Multilink debugger cable used for this project is:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=USBMULTILINKBDM&parentCode=S12XD&fpsp=1

http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=USBMULTILINKBDME-ND

References:

http://www.holtic.com/

REVISION HISTORY

P/N	Rev	Date	Description of Change
AN-166	NEW	08/12/11	Initial Release