



HI-3220 ARINC 429
DATA MANAGEMENT ENGINE
16 x RECEIVERS
8 x TRANSMITTERS

ADK-3220 Application Development Kit
Users Guide

April 25, 2018

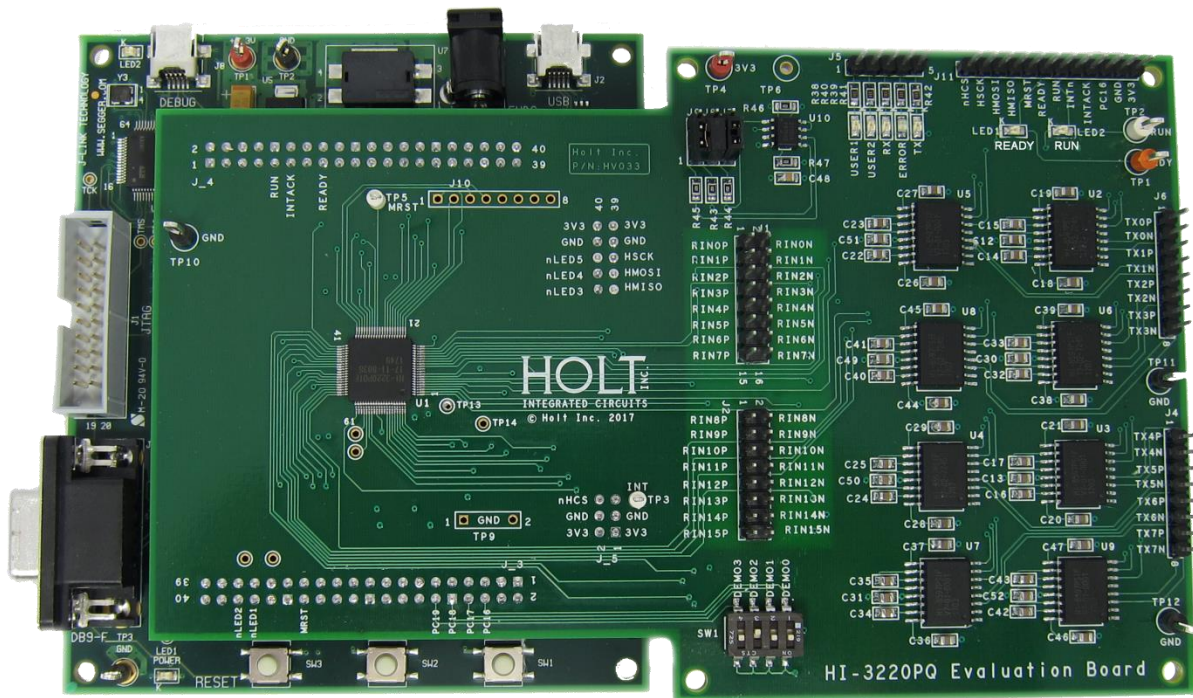
REVISION HISTORY

Revision	Date	Description of Change
AN-3220, Rev. New	01-26-18	Initial Release
A	04-25-18	Correct Op Code typo on p. 31. 0xEE should be 0xE0. Pins J7 and J9 were erroneously reversed in Mode Table on p. 7.

Introduction

The Holt ADK-3220 Evaluation board demonstrates the broad feature set of Holt’s ARINC 429 Data Management Engine, with sixteen ARINC 429 Receivers and eight digital Transmitters integrated into a single IC. The HI-3220 is the successor to the HI-3200 and HI-3210 with expanded channels and new feature enhancements designed to increase host communication efficiency. The HI-3220 receiver channels features DO-160G lightning protection. The HI-3200 has a maximum SPI rate of 40MHz.

This guide describes how to set up and run the board. Additional support material and all required project software are found in the included Holt USB memory stick. A version of the demonstration software is pre-programmed into the microcontroller flash; the board is operational right out of the box without installing or running the provided software development tools.

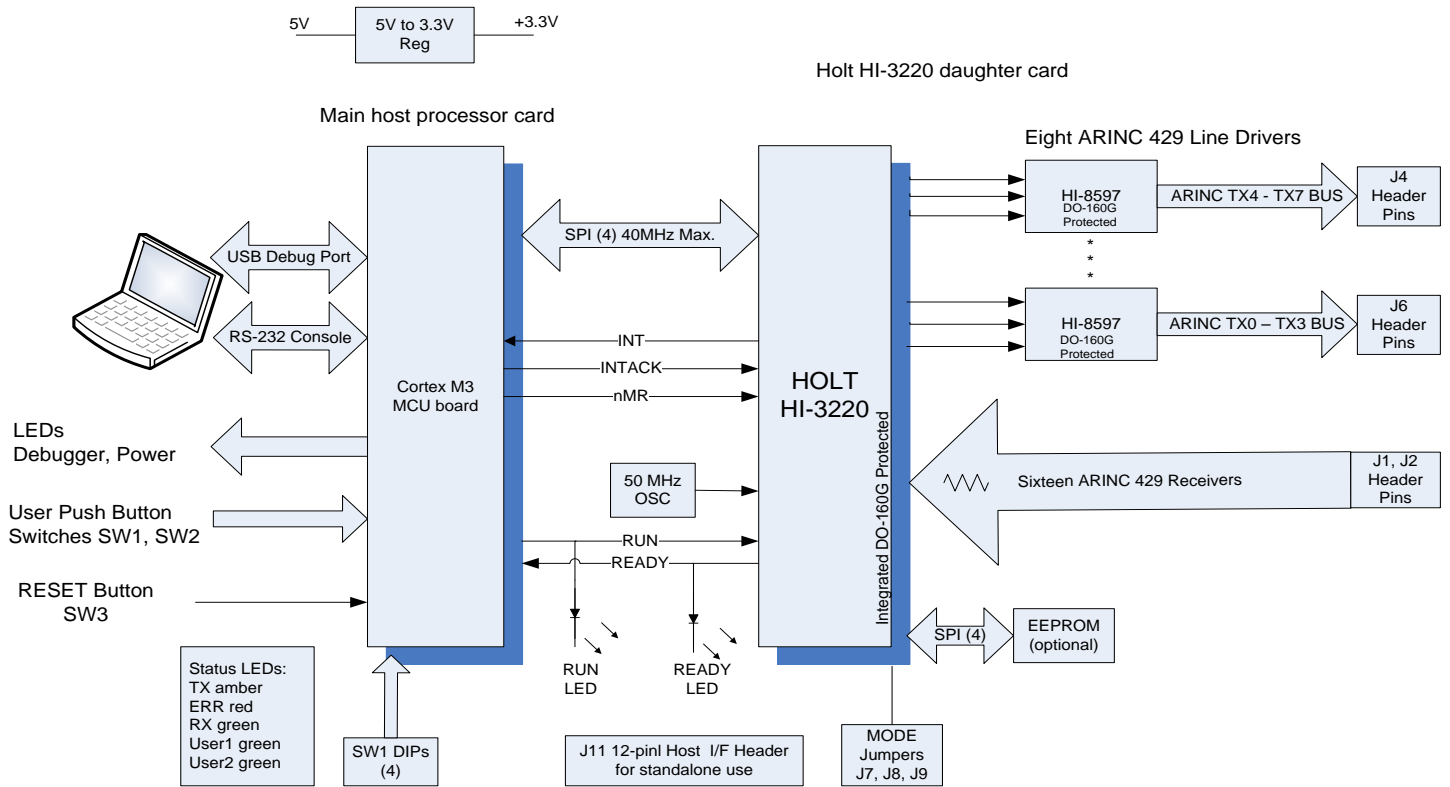


ADK-3220 CTX Evaluation Board, mounted on the ARM Cortex M3 MCU Board

Evaluation Kit Contents

- This User Guide.
- Holt HI-3220 Project Software and Documentation USB memory stick.
- IAR Systems Embedded Workbench® for ARM (32KB KickStart) on USB memory stick.
- Plug-in 5V DC power adapter.
- USB mini debug interface cable.
- RS-232 serial cable, DB-9M to DB-9F for console I/O using a connected computer.
- USB to serial dongle adapter.
- 2-board assembly comprised of:
 - Upper HI-3220 evaluation board. DIP switches configure board operation.
 - Lower MCU board with (Microchip/Atmel) ARM Cortex M3 16-/32-bit microprocessor, debug interface and regulated 3.3VDC power supply

Hardware Block Diagram



HI-322x Highlighted Feature Differences Compared to HI-3200/3210 Family

- Increased SPI clock rate to 40MHz.
- Up to Sixteen integrated ARINC 429 Receiver channels (eight more than HI-3200/10 family) with DO-160G level-3 lightning protection. Lightning protection available on parts with integrated on-chip receivers (3220, 3221 and 3222).
- Up to Eight ARINC 429 Transmit channels (four more than HI-3200/10 family).
- Receiver 64 word deep FIFO's increased from 32.
- Transmit 32 word deep FIFO's.
- Transmit and Receive FIFO count registers enables efficient host reading.
- Transmit and Receive FIFO threshold count interrupt registers.
- ARINC Transmit and Receive 50KHz rate support in addition to standard ARINC 429 12.5 KHz and 100 KHz rates.
- ARINC Transmit TRISTATE control bit for enabling Hi-Z mode in certain Line Drivers.
- Transmit scheduler feature improvements:
 - Simplified Transmit scheduler Op Codes.
 - Packet Timer period and offset control.
 - Delay Op Code.
 - Transmit scheduler 1ms prescaler option.
 - Transmit Watch-Dog Timer function.
- Ability to read the Mode status bits [7:5] in register 0x807A indicating the state of the Mode0-2 pins latched after a reset.

Push-buttons

Push-button Switches (on main CM3 board)	DESCRIPTION
SW1	Transmits a Test message on channels TX5, TX6 and TX7. TX5 = Low Speed (12.5KHz) TX6 = Mid Speed (50KHz) TX7 = High Speed (100KHz)
SW2	Simple SPI sequence test sequence – useful for trouble shooting the SPI interface. See section “ HI-3220 SPI basics for trouble shooting ” near the end of this document.

DIP SWITCHES

SWITCH	DEFAULT	DESCRIPTION
SW 1	OFF	SPI frequency select. OFF = 24MHz. ON = 16MHz. Checked at power up.
SW 2	ON	Open = Logs TX descriptor table values to the console.
SW 3	ON	Available to user
SW 4	ON	Available to user.

AN-3220

Jumper Settings for EEPROM auto-initialization (optional feature)

Factory default is Mode 0. See RESET and START-UP MODES in the data sheet for the list of self-checks the HI-3220 performs in each mode.

MODE	M2 J9	M1 J8	M0 J7	EEPROM Init
0	0	0	0	NO
1	0	0	1	YES
2	0	1	0	No
3	0	1	1	YES
4	1	0	0	NO
5	1	0	1	NO
6	1	1	0	RAM BIST ENA

HI-3220 supply current measurement test points TP13 and TP14

To measure the HI-3220 supply current remove R35 zero-ohm resistor and insert a current meter between TP13 and TP14. When READY is high there is addition current through LED1 so remove R33 or subtract the LED1 current (~3.6mA) from the measured current.

External Host interface to the HI-3220 daughter board.

The HI-3220 daughter board can be separated from the MCU board and connected to a user FPGA or MCU host for quick prototyping purposes. All essential interface signals are provided on J11 12-pin header connector. The Demo Mode switches and LEDs are not included, but they are accessible on J3 and J5 header connectors, if needed.

J11 Pin	SIGNAL	J11 Pin	SIGNAL
1	nHCS (SPI)	7	RUN
2	HCK (SPI)	8	INTn
3	HMOSI (SPI)	9	INTACK
4	HMISO (SPI)	10	SW1 -1 DIP
5	/MRST	11	GND
6	READY	12	3.3 VDD

LED indicators

LEDS	FUNCTION	TYPICAL USAGE
LED1	READY	On when READY is high (connected to READY pin)
LED2	RUN	On when RUN is high (connected to the RUN pin)

AN-3220

LED3	RX	Host controlled – Typically On when reading FIFO messages.
LED4	USER2	Host controlled - Available to user
LED5	USER1	Host controlled - Flashes ONLY in main idle loop.
LED6	ERROR	Host controlled - for flagging errors , like EEPROM write errors
LED7	TX	Host controlled - Typically On when transmitting

Getting Started

The first section of this user guide provides instructions how to set up the demo board and run the built-in demo programs and utilities that are pre-programmed into the Microchip/Atmel ARM Cortex M3 MCU. The next section instructs how to install the IAR *EWARM* IDE and the Holt demonstration program. The supplied demo program on the USB memory stick is the same version already programmed in MCU flash memory.

Hardware Design Overview

Refer to the end of this guide for separate schematic diagrams and bills of material for the upper daughter board and lower MCU board.

The detachable daughter board can be separated from the lower MCU board for connection to a user-supplied alternate MCU or FPGA board. The inter-board headers are located on 0.1" (2.54 mm) grid for compatibility with generic prototyping boards. All host interface signals go through the inter-board headers. See the External Host interface header pin description for the pin list in the previous section.

The lower ARM Cortex M3 board is based on the flash-programmable Microchip /Atmel AT91SAM3U-EK MCU. A 4-signal Serial Peripheral Interface (SPI) connects to the HI-3220 DUT. A UART-based serial port provides RS-232 console I/O (optional). An uncommitted USB 2.0 port is available for future expansion. Two pushbuttons are available for software interaction. A RESET pushbutton resets the ARM microprocessor, which in turn controls the DUT Master Reset signal.

The ARM Cortex M3 board includes "J-Link On Board" debug interface, licensed from www.segger.com, providing out-of-box readiness without having to buy a costly JTAG debug cable. The kit includes a mini USB cable for connecting the board's debug interface to your computer. (For users already owning an ARM debug interface with ribbon-cable connector, an ARM-standard 2x10 debug connector provides debug connectivity. In this case, jumper JP2 on the bottom of the lower board should be soldered closed to disable "J-Link On Board").

Initial Set Up

Before attempting to run the demo programs, a suitable PC is required with serial COM support so the demo programs and utilities can be invoked by entering commands on the console menu. This demo program is already flashed into the Cortex M3 MCU and operates out of the box – no IDE flash programming is required to run the demo programs.

1. Your PC will need a serial (COM) port and a “terminal emulation” program like TeraTerm. Most computers no longer have RS232 com ports so will require a serial-to-USB adapter, supplied with the ADK. Connect this to the computer’s USB port and the 9 pin connector to the ADK board.
2. If using Windows 2000 or Windows XP, you can use HyperTerminal for terminal emulation. Open HyperTerminal by clicking **Start** then **All Programs**; click the Windows **Accessories** then **Communications** program group. Double-click HyperTerminal to run it. Skip the next paragraph.

If using Vista, Windows 7 or Windows 10...

HyperTerminal is not included with these versions of Windows. Install the free open-source terminal emulation program, *TeraTerm 4.71*, by running the provided teraterm-4.71.exe installer program from the Holt CD. Accept the license agreement stating redistribution is permitted provided that copyright notice is retained. The notice can be displayed from the *TeraTerm* window by clicking **Help** then clicking **About TeraTerm**. Continuing to install...

- Accept the default install destination and click **Next**.
- At the Select Components screen, unselect all options except Additional Plug-in = TTXResizeMenu and click **Next**.
- Select the installed language, then click **Next**.
- Accept the default Start Menu folder, then click **Next**.
- Select any desired shortcuts, then click **Next**.
- At the Ready to Install screen, click **Install**.

Run the *TeraTerm* program. At the **New Connection** screen, select **(x)Serial** and choose the selected COM port. Click **Setup** then **Serial Port** to open the serial port setup window. Choose these settings: Baud Rate: 115200, Data: 8 bits, Parity: none, Stop: 1 bit, Flow Control: none.

3. Plug-in the provided 5V DC power supply and connect the cable to the power input jack on the lower circuit board. If *TeraTerm* is running and configured correctly, the command menu below should appear in the console window. This menu appears whenever board power is applied, or when the RESET pushbutton is pressed. After verifying correct *TeraTerm* communication with the evaluation board, the terminal set up can be saved by clicking **Setup** then **Save Setup**.

- At power up, all the LED's will briefly flash. The READY LED should remain ON and the RUN LED should be OFF. The green USER1 LED flashes at 1.2 Hz when the program is in the main idle loop.

Getting Started with Demos

After powering up the board the menu of commands are displayed on the console.

```

-----
Holt Integrated Circuits HI-3220 Demo Program Ver. 1.0
  Compiled: Jan 26 2018 08:27:43
-----

DIP switches: 01
SPI 24MHz
Waiting for READY high after reset...
MSR = 20 Expected value is 20
READY OK
MODE Pins read: 0
MODES 1 or 3 auto-initializes from EEPROM - must have been programmed first or will fault

*****
Holt Integrated Circuits HI-3220 Demo Program Ver. 1.0
  Compiled: Jan 26 2018 08:27:42
*****

----- ADK-3220 DEMOS -----
Press '0' - RX RAM RECEPTION LOOPBACK
Press '1' - RX RAM RECEPTION NO LOOPBACK
Press '2' - RX FIFO and TX FIFO Direct
Press '3' - TX FIFO Direct Immediate
Press '4' - TX0 Simple Transmit Scheduler-continuous
Press '5' - RX0 -> TX0 Repeater
Press '6' - RX0 + RX1 -> TX0 Concentrator
Press '7' - SUB MENU: 1-One Time Transmission, 2-PacketTimer, 3-RX&TXDemo
                4- Transmit using all 8 schedulers
Press '8' - Enables RX0-RX15 receivers, use with command '9'
Press '9' - Reads and Displays all RX channels FIFOs
Press 'p' - Demo configuration for EEPROM demo
Press 'P' - Programs auto-initization EEPROM
Press 'k' - Transmit a message on TX2 and TX3

----- UTILITIES -----
Press 'a' - Enter Watch Window Address
Press 'c' - Clears most device memory 0-0x8088
Press 'd' - Display selected RX RAM Memory blocks 0-0x3FFF, q=quit
Press 'e' - Display Receiver FIFO look-up label and interrupt look-up tables Memory
Press 'h' - Display this help menu
Press 'i' - Set interrupts SUB-MENU
Press 'I' - Set Interrupt regs for testing
Press 'l' - Toggle the LOOPBACK register from 00 <-> 0xFF
Press 'L' - Demonstrate setting and clearing Label acceptance table memory
Press 'm' - Write byte to specified Address
Press 'M' - Write byte(s) to memory Range
Press 'O and o' - WDT Demo, init and refresh
Press 'r' - Toggle RUN pin state
Press 'R' - 3220 Software Reset
Press 't' - Display ALL Transmit Tables 0x4000-0x67FF, q=quit
Press 'w' - Display Device Memory from MAP
Press 'x' - Transmits number of mesgs on given channel
Press 'Space Bar' - Display Selected Registers
=====

Press 'H' for this help menu, or press any valid menu key. >>

```

Demonstration Programs and Utilities

It is highly recommended to first review the HI-3220 data sheet to gain basic understanding of the device so you get the most benefit from the demos and utilities described in this document.

Nearly all menu commands entered on the console instruct the ARM MCU to issue SPI Op Codes to write or read data in the HI-3220. The host configures the HI-3220 by writing to various Control Registers using the SPI interface along with other related system registers in space 0x8000 - 0x8087. Access to registers or memory locations are enabled once the READY pin is asserted high after a master reset. Message data, transmit schedulers, RX FIFO enable, RX interrupt enable tables reside from location 0x0000 to 0x8000. See the HI-3220 Memory Map in page 11 of the data sheet. Once the host configures the device and loads necessary data tables, the device is enabled to receive or transmit data when the RUN pin is set high. The RUN input pin of the HI-3220 is controlled by the MCU GPIO pin and is toggled by the ‘r’ command. When the RUN pin is high the green RUN status LED is ON. More about the RUN pin later.

In addition to the demos there is an assortment of SPI utilities which allow viewing and modification of registers or memory ranges, and setting/clearing single bits in the RX interrupt and RX FIFO Enable MAP tables. These are very useful debugging utilities and are provided since traditional IDE debuggers cannot access memory through a SPI interface. Note: By default, all SPI utilities use hex format for display of addresses or data without showing the traditional “0x” in front of the value.

Demonstrations and Examples

The first set of demo programs 0-6 are modeled by the Example diagrams 1-5 shown in the data sheet on pages 3, 4 and 5. In the demo descriptions, the documentation will typically refer to these methods shown in the 3rd column. The demos demonstrate one or two of these methods to transmit and receive data. The key demos listed below demonstrate the essential features of the HI-3220. Entering a number in the console menu executes the corresponding demo listed.

DEMO NUMBER	DEMO DESCRIPTION	NOTES
Example 0	On-chip RAM reception – Loopback.	The Transmitter by Direct FIFO method is also used.
Example 1	On-chip RAM reception.	The Transmitter by Direct FIFO method is also used.
Example 2	Data reception using on-chip FIFO's.	RX FIFO Reception
Example 3	TX FIFO Data transmission direct from MCU.	Transmit Direct by FIFO loading

AN-3220

Example 4	TX Simple Data transmission on-chip schedulers. Requires Descriptor table(s)	Transmit scheduler(s)
Example 5	RX0 → TX0 Repeater	Messages received on RX0 are retransmitted on TX0
Example 6	RX0 + RX1 → TX0 Concentrator	Messages received on RX0 and RX1 are <u>both</u> retransmitted on TX0
Example 7	Sub Menu commands	See descriptions below
Example 8	This is a Demo utility that enables all sixteen receivers used in some the other demos.	Configures all 16 Receivers for reception.
Example 9	This is a Demo utility reads any RX FIFO messages and displays them on the console used in some of the other demos.	

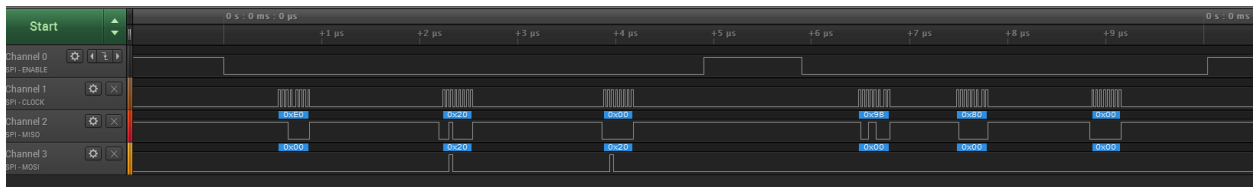
AN-3220

Two pushbutton switches on the bottom board provide two utility functions.

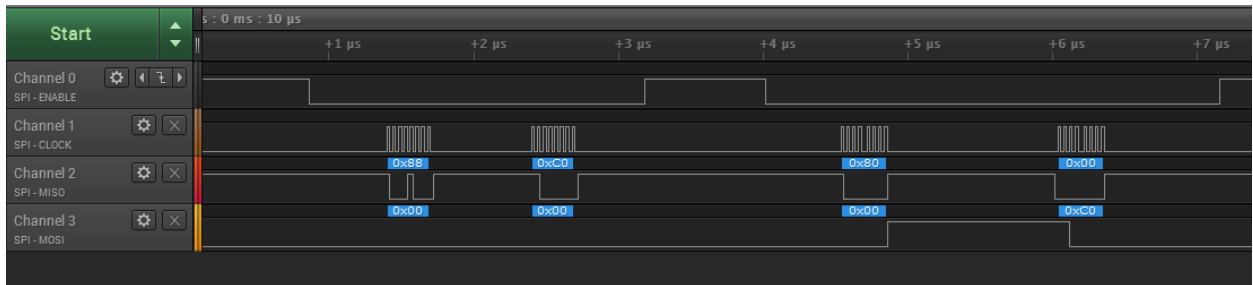
SW1 Push-button - Configures the MCR and TX5, TX6 and TX7 Control Registers and transmits a single message on those three transmitter channels. TX5 transmits at Low-Speed, TX6 at 50K speed and TX transmits at Hi-Speed. The green TX LED will briefly flash.

SW2 Push-button - Generates four essential SPI sequences that demonstrate proper SPI signaling to communicate with the HI-3220. For a detailed look at the SPI signaling refer to the end of this document.

Read Register MSR (0x02) returns value 0x20, Write 0x8000 to the MAP register.



Write 0xC0 to the MAP address location (0x8000), Read byte at MAP location returns 0xC0.

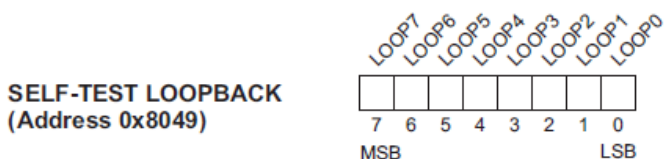


Demo Descriptions

Demo 0:

Press the '0' key for RAM Reception - LOOPBACK.

Demo 0 programs the LOOPBACK register with 0xFF so all RX channels are configured in LOOPBACK mode. Messages are loaded into Receive Data memory organized by label value. Each channel 0-15 has 256 4-byte locations for each message organized by label value. TX0 - TX7 transmitters are used to transmit the test data using transmit direct method by loading the Transmit FIFO's with message data. All transmitters are configured in high-speed and Parity ON is enabled for RX0 and RX1. All sixteen Receivers are similarly configured. Initially the transmit data is set for Label 00 and data 00 45 67. ARINC 429 data is transmitted to the eight channels using the Transmit Direct method (example 3). After each transmitter FIFO is loaded the label and byte 1 are incremented by one. In LOOPBACK mode, each Transmitter is mapped to two consecutive Receivers as shown below.



Bit	Name	R/W	Default	Description
7	LOOP7	R/W	0	This bit is set to "1" to loop-back transmit channel 7 to receivers 14 and 15
6	LOOP6	R/W	0	This bit is set to "1" to loop-back transmit channel 6 to receivers 12 and 13
5	LOOP5	R/W	0	This bit is set to "1" to loop-back transmit channel 5 to receivers 10 and 11
4	LOOP4	R/W	0	This bit is set to "1" to loop-back transmit channel 4 to receivers 8 and 9
3	LOOP3	R/W	0	This bit is set to "1" to loop-back transmit channel 3 to receivers 6 and 7
2	LOOP2	R/W	0	This bit is set to "1" to loop-back transmit channel 2 to receivers 4 and 5
1	LOOP1	R/W	0	This bit is set to "1" to loop-back transmit channel 1 to receivers 2 and 3
0	LOOP0	R/W	0	This bit is set to "1" to loop-back transmit channel 0 to receivers 0 and 1

For each '0' key press, this output will be displayed on the console. Three interrupt flags and the system registers are displayed before the messages for each channel 0-15 for debug information.

```
Quit> 0
ARINC RAM Reception LOOPBACK
Loop Back Mode

PIR   REG 0x8004   = 0x04
RPIR1 REG 0x8006   = 0xFF
RPIRH REG 0x8008   = 0xFF
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:8000 C0 C0 30 00 00 00 00 00 00 00 00 00 FF FF FF
MAP:8010 00 00 01 01 02 02 03 03 04 04 05 05 06 06 07 07
MAP:8020 80 80 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 (RX Control Registers)
MAP:8030 80 80 80 80 80 80 80 80 00 00 00 00 00 00 00 (TX Control Registers)
MAP:8040 00 00 00 00 00 00 00 00 00 00 FF F8 FF FF 00 00 00 (LOOPBACK Reg 0x8049=0xFF)
MAP:8050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8060 00 00 00 00 00 00 00 00 01 01 01 01 01 01 01 01
MAP:8070 01 01 01 01 01 01 01 01 00 00 00 00 00 00 FF 00
```

```
** RX messages by polling **
RX-CH0: Label:00 00 45 67
RX-CH1: Label:00 00 45 67
RX-CH2: Label:01 01 45 E7
RX-CH3: Label:01 01 45 E7
RX-CH4: Label:02 02 45 E7
RX-CH5: Label:02 02 45 E7
RX-CH6: Label:03 03 45 E7
RX-CH7: Label:03 03 45 E7
RX-CH8: Label:04 04 45 E7
RX-CH9: Label:04 04 45 E7
RX-CH10: Label:05 05 45 E7
RX-CH11: Label:05 05 45 E7
RX-CH12: Label:06 06 45 E7
RX-CH13: Label:06 06 45 E7
RX-CH14: Label:07 07 45 E7
RX-CH15: Label:07 07 45 E7
Quit>
```

Optionally, instead of reading the Receiver data by RAM access, the data can be read from a FIFO. This is demonstrated in Demo 2.

By default, the C program is compiled for Polling using the macro `INTERRUPT_MESG_ENA`. A simple interrupt method to read the message is provided by changing this macro to a 1 in the `3220_initialization.h` file which then causes the messages to be read and displayed in the interrupt handler.

Demo 1:

Press the '1' key for RAM Reception - normal mode.

Demo 1 is the same demo as Demo 0 except the LOOPBACK feature is turned OFF (LOOPBACK register 0x8049 = 0x00). An oscilloscope can be used to view the ARINC transmissions on J4 or J6 header pins. For example J6 pin 1 is TX0AOUT and pin 2 is TX0BOUT. Without jumper wires connected between a Transmitter and a Receiver, only the register data is displayed on the console. Optionally connect an external ARINC 429 high speed source to any of the Receiver input channels 0-15 on the header connectors. RX channels 0-7 are available on J1 and channel 8-17 are on J2.

To set up a single Transmit channel to be received by a Receiver channel connect the following using a pair of clip leads. Connect TX0 outputs to RX0 inputs and verify that data is displayed on the console when the '1' key is pressed. Optionally connect more channels or an external ARINC 429 source and press the '1' key again to see the data.

J6-1 TX0AOUT	Jumper to	J1-1 RINOP
J6-2 TX0BOUT	Jumper to	J1-2 RINON

AN-3220

```
> 1
ARINC RAM Reception NO LOOPBACK
Normal Mode

PIR REG 0x8004 = 0x04
RPIR1 REG 0x8006 = 0x02
RPIRH REG 0x8008 = 0x75
      0 1 2 3 4 5 6 7 8 9 A B C D E F
MAP:8000 C0 C0 30 00 00 00 00 00 00 00 00 00 00 02 75 FF
MAP:8010 00 03 00 00 00 00 00 00 04 00 05 00 06 06 07 00
MAP:8020 80 80 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
MAP:8030 80 80 80 80 80 80 80 80 00 00 00 00 00 00 00 00
MAP:8040 00 00 00 00 00 00 00 00 00 00 00 00 F8 FF FF 00 00 00
MAP:8050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8060 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00
MAP:8070 01 00 01 00 01 01 01 00 00 00 00 00 00 00 00 FF 00

** RX messages by polling **
RX-CH0: Label:00 00 45 67
Quit>
```

Demo 2:

Press the '2' key for RX FIFO Reception and TX Direct Transmit.

This demonstrates Receiver FIFO reception on all sixteen Receivers.

Each Transmit channel TX0-TX7 transmits 32 messages to supply message data to the RX0-RX15 Receivers. Since LOOPBACK was demonstrated in Demo 0 and 1 this demo relies on external clip-lead connections. A pair of clip-leads is required for each channel. For example to connect TX0 to RX0 and TX7 to RX15 make the following connections.

J6-1 TX0AOUT	Jumper to	J1-1 RIN0P
J6-2 TX0BOUT	Jumper to	J1-2 RIN0N
J4-7 TX7AOUT	Jumper to	J2-15 RIN15P
J4-8 TX7BOUT	Jumper to	J2-16 RIN15N

A data pattern function creates patterns as follows for 32 messages each which are loaded into the Transmit FIFO's. The first two bytes (label and Byte1) are loaded with the label. Byte2 is loaded with increment data starting with value 0x00. Byte3 is the complement of Byte2.

Each Receiver channel features a 64 message deep FIFO. The RX FIFO read section of the software first reads the message count in each RX FIFO channel then it calls a function to read that many messages. It repeats this for all sixteen Receiver channels. See the demo code for exact details.

The console display should look like this after the connections are made and the '2' key is pressed.

```
RX 0:0  00 00 00 FF
RX 0:1  00 00 01 FE
      *
      *
      *
RX 0:30 00 00 1E E1
RX 0:31 00 00 1F E0

RX 15:0 07 07 00 FF
RX 15:1 07 07 01 FE
      *
      *
      *
RX 15:30 07 07 1E E1
RX 15:31 07 07 1F E0
```

Demo 3:

Press '3' to transmit a message on TX2 and TX3 using transmission directly by CPU.

This method has already been demonstrated in the previous demos 0-2. Transmission by Direct method is when the Transmit FIFO is loaded with data and the RUN pin is set high. This causes the transmission of any data in the FIFO's for that channel until the FIFO is empty. Pressing 'SW3' push-button also uses this method to transmit three messages, one message each on TX5, TX6 and TX7. Demo 'k' also uses this method to transmit a message on TX2 and TX3.

Transmit Scheduler demos

The next set of Demos use the Transmit Scheduler features of the HI-3220 to accomplish a variety of capabilities. Each Transmit channel TX0-TX7 has a transmit controller and supporting descriptor table space in memory starting at address 0x4000. For channels 0 and 1 either 128 or 256 message descriptors can be loaded into the descriptor memory space. Channels 2-7 are fixed at 128 messages. Each descriptor consists of eight bytes for an Op Code, timer values and message data. See the data sheet for more details for how the Transmit Scheduler works. Once the descriptors are initialized and Control Register bits are properly set, the scheduler runs when the RUN pin is set high. To stop the scheduler set the RUN pin low. Use command 'r' to toggle the RUN pin high or low in these demos.

Some examples what each a Transmit Scheduler can be used for:

1. Transmit up to 128 or 256 immediate messages.
2. Transmit messages automatically conditionally or unconditionally if message is received from any Receiver channel and by ARINC label value.
3. Transmit messages the same as item 2 above, but optionally reinsert ARINC SDI bits (ARINC bits 9 and 10).
4. Autonomous operation once the auto-initialization EEPROM is programmed.

5. One-shot or continuous transmission of the number of programmed message descriptors until the end of the table is reached or the End of Sequence value 0x00 is encountered by the Scheduler.
6. The Transmit Scheduler message auto routing capabilities allows the device to be configured as a Data Concentrator or Repeater.

Demo 4:

Press '4' for TX0 Simple Scheduler – continuous.

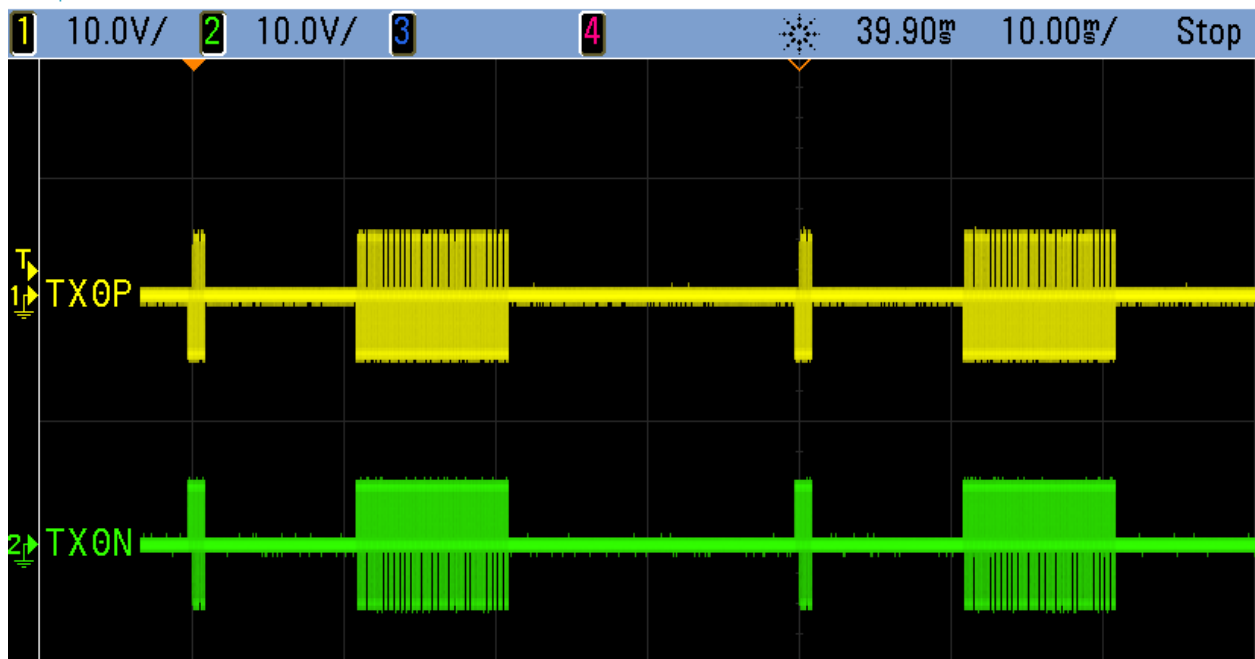
This demo initializes the Transmit TX0 descriptor table with 31 messages with the Immediate Op Code. After the 3rd message transmits, a 10ms delay Op Code delays transmission before the remaining messages transmit. All 31 messages are the same except byte3 that increments, Label 00, byte1: 0x00, byte2 increment data, and byte3 =0x04.

After each message is loaded in the descriptor table, the label is incremented for the next descriptor. After the program loads all 32 descriptors, an End of Sequence Op Code is written in the next address with 0x00. The repetition rate is set for 40ms.

After pressing '4', the demo quickly returns to the main console menu.

Press 'r' to set the RUN pin high to start the scheduler, the first three messages transmits followed by a 10ms delay then the remaining messages transmit. This occurs every 40ms. If 'r' is pressed again, RUN goes low and transmission stops and if 'r' is pressed again the RUN pin is set high again and transmissions resume.

View the ARINC TX0 channel transmissions on J6 pin 1 and 2 with an oscilloscope.



To read back the data and display it on the console using the FIFO Reception method, first connect the TX0 transmitter outputs to the RX0 Receiver inputs as follows.

J6-1 TXOAOUT	Jumper to	J2-1 RINOP
J6-2 TXOBOUT	Jumper to	J2-2 RINON

Press '8' to initialize Receivers 0-15 then press 'r' to start transmission. Press 'r' again to stop transmission.

Press '9' to display 64 messages on the console. There are 64 messages because that is the Receiver FIFO depth. Probably several 64-message groups were transmitted between the first and second 'r' key presses, and the FIFO overflowed. When the Receive FIFO fills, a subsequent message is loaded in the FIFO overwriting the oldest stored message.

You can also leave RUN high and randomly press '9' to display the FIFO messages in this case the first displayed message will unlikely be the first message in the descriptor table.

Commands '8' and '9' from the console menu can be used anytime to configure the RX Receivers and display the RX FIFO data.

Repeater Demo

Press '5' for RX0 -> TX0 Repeater.

This demo sets up a single descriptor in the TX0 scheduler table to perform a Repeater function and returns control back to the console main loop. Pressing 'r' sets the RUN pin high to start the scheduler. Any message received on Receiver 0 automatically re-transmits the message on TX0 without host intervention. Optionally, the host can receive the message if desired, using RAM Reception or FIFO Reception.

The descriptor is configured to reference RX0 channel, label 0 message in RAM memory (0x0000).

This example uses an IDC Op Code for byte-1, the re-transmitted message will have the label value stored in Byte5. Bytes 2, 3 and 4 are indexed and come from RX0 channel where the label 0x00 message data is stored, but actually the first byte is a Status Byte in this location. Review Demo 5 code in `3220Demo.c` to view the C code configuration steps.

Indexed Conditional Op Codes use NEWTX0 - NEWTX3 bits in the Received Data Status Bytes. This Status Byte is the first byte in Received data memory (See page 19 of the data sheet) that is loaded for every received message. When a message is received, all the bits in this Status Byte register are set. This is how

AN-3220

the scheduler detects when a new message is received. If a descriptor references any of the data from this block, the corresponding NEWTXx bit is cleared depending on the Transmitter channel (TX0-TX7).

The single Transmit descriptor consisting of eight bytes shown below is loaded into starting location 0x4000 since that is the first location of the TX0 descriptor table. These values are set in arrays which makes them easy to alter in C code for experimentation.

Descriptor Bytes 1-8

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Op Code	RX Label	PTP/Delay	PTO	ARINC Label	ARINC Byte1	ARINC Byte3	ARINC Byte3
IDC (0101)	0x02	0x00	0x00	0x02	N/A	N/A	N/A

For a repeater, PRESCALE bit 6 is set in the TX0 Control Register. This sets the repetition rate to 1ms to minimize latency between message arrival on the RX0 Receiver to when the Scheduler retransmits the message on TX0.

To view the descriptor bytes in memory, use the 'a' command utility and enter address 4000. The descriptor memory should appear with the following data (only partially shown):

```
      0 1 2 3 4 5 6 7 8 9 A B C D E F
MAP:4000 50 02 00 00 02 00 00 00 00 00 00 00 00 00 00 00
                ^--- EOS
```

For example:

```
RX0 Incoming message: 00 02 34 56
TX0 Repeated message: 00 02 34 56
```

To change the transmitted label, simply change the label (Byte 5) to any desired value in the software. In the demo program, descriptor bytes are stored in eight byte arrays[]. Experimentation can be performed by changing the descriptor byte values in the array and rebuilding the project or by altering bytes by using the SPI utility command 'm' which allows writing a byte to any address entered on the console. View the result with the SPI utility command 'a' to set the starting address and display a block of addresses and corresponding data values.

For this demonstration, connect two clip-leads to make an external connection between TX2 outputs and RX0 inputs (similar to the previous demos). When the 'k' key is pressed, a message will transmit on TX2 so this will be the source or input message the Repeater retransmits on TX0.

Steps:

AN-3220

Press '5' to execute the Repeater demo.
Press 'r' to run scheduler (RUN high)
Press 'k' to transmit test messages on TX2 and TX3 (TX3 is ignored for this demo.)
Press '9' to display the message.

To see the repeated transmission, connect another pair of clip-leads from the TX0 output pins to the RX3 channel inputs. This allows read back and retransmitted message display on TX0 on the console.

Results:

> 5

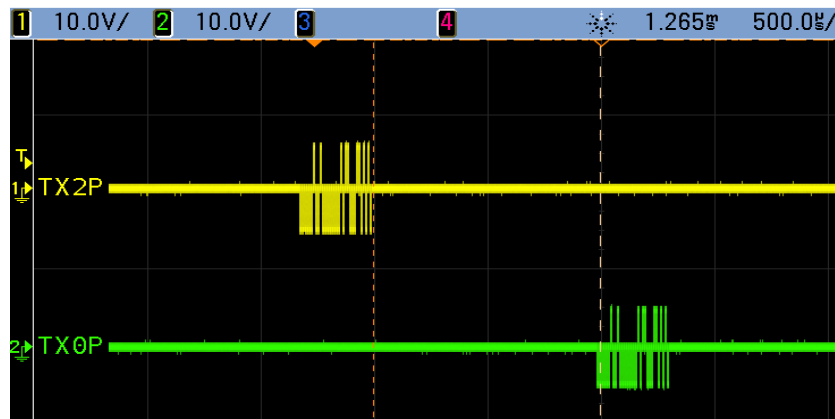
Press '6' for Repeater Demo
Press '8' to enable Receivers
Press 'r' to set RUN high which runs the scheduler
Press 'k' to transmit TX2 TX3 test messages – Transmits message 0x02 0x02 0x34 0x56 on TX2
Press '9' to read the FIFO data and display it on the console

Displayed result:

```
msgs: 1
RX 0:0  02 02 34 56
msgs: 1
RX 3:0  02 02 34 56 Elapsed time 0.007210 >
```

Each time 'k' is pressed, a message is transmitted on TX2 and this is received by RX0 and the scheduler retransmits the message on TX0.

The scope capture below shows the TX2 transmission on top and the retransmitted message on TX0 below. The Scheduler 1ms timer is asynchronous relative to messages, so the worst case latency is shown below at 1ms in this case. Pressing 'k' several times will reveal delays anywhere between the two vertical cursor lines.



Demo 6:

AN-3220

Press '6' for RX0 + RX1 -> TX0 Concentrator Demo.

The concentrator demo receives messages on two Receiver channels (RX0 and RX1) and retransmits them both on TX0 transmitter.

This demo sets up two descriptors in the TX0 scheduler table. The first descriptor references the RX0 channel message and the second one references the RX1 message (both Label 0). After pressing '6', control returns back to the main console menu. Press 'r' to set the RUN pin high, starting the scheduler. Any new messages received on Receiver 0 or Receiver 1 will automatically re-transmit to TX0 with no host intervention. Optionally, the host can receive the messages if desired, using either RAM access or FIFO Reception. This is very similar to the Repeater demo except two descriptors are loaded into the TX0 descriptor table at location 0x4000.

View the descriptor table at location 0x4000 (using the 'a' command):

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:4000 50 02 00 00 11 00 00 00 51 03 00 00 22 00 00 00
MAP:4010 00 (EOS)
```

The 'k' command is again used here to transmit test messages on TX2 and TX3 outputs.

TX2 and TX3 outputs should be connected to RX0 and RX1 respectively. And finally, jumper TX0 to RX3 to view TX0 output received by RX3 displayed on the console

TX2 transmits 00 02 34 56 → Received by RX0

TX3 transmits 00 03 78 9A → Received by RX1

Results:

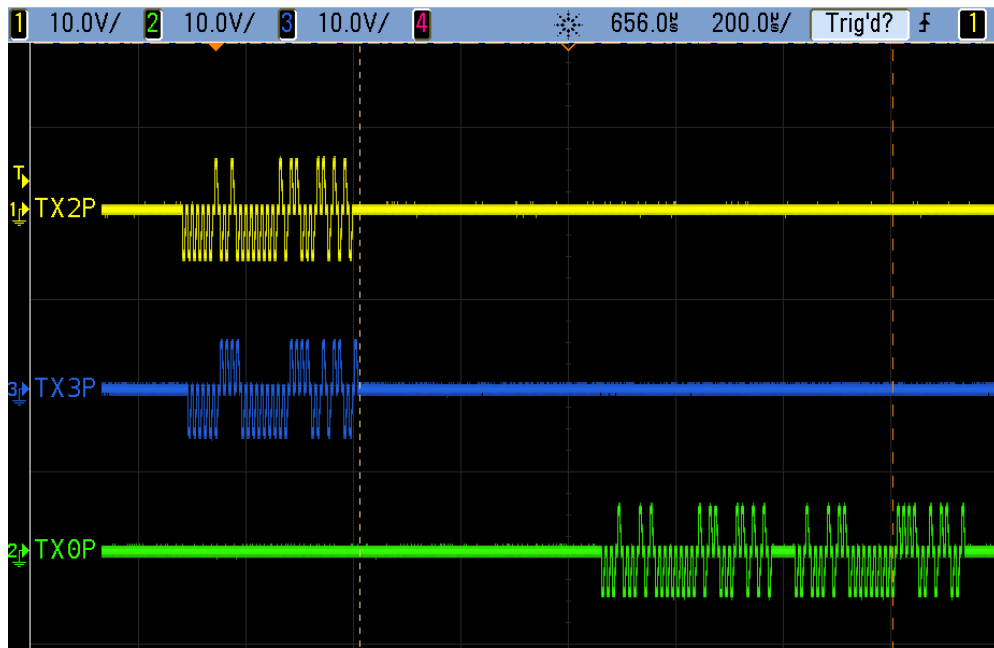
```
> 6
Concentrator Demo
Press 8 to enable the receivers
Press r to set RUN high to run scheduler
Press k to transmit TX2 TX3 test messages
Press 9 to read the FIFO data and display it on the console

      msgs: 1
      RX 0:0  02 02 34 56
      msgs: 1
      RX 1:0  03 03 78 9A
      msgs: 2
      RX 3:0  11 02 34 D6
      RX 3:1  22 03 78 9A
```

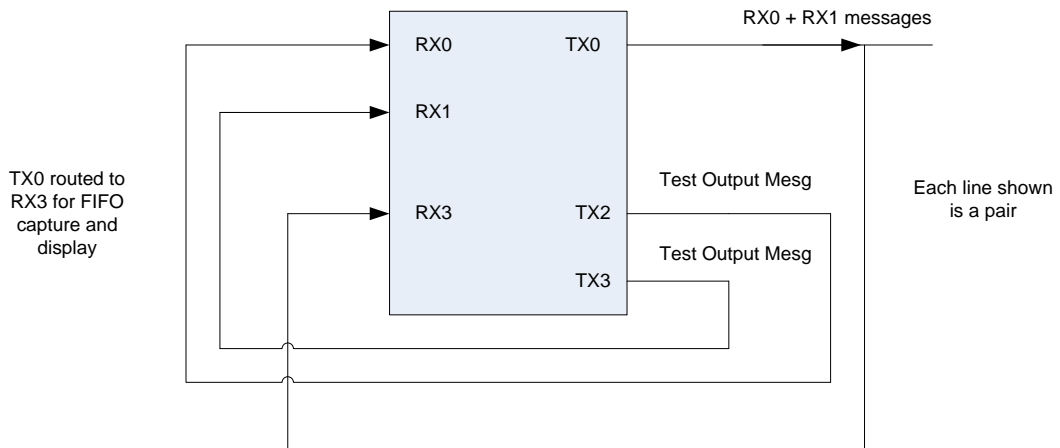
A few things to note here: The label values were changed to 0x11 and 0x22 in the descriptor tables for this demo so those are the new transmitted label values the concentrator transmits. The RX3 receiver is monitoring TX0 output and receives both messages from RX0 and RX1 and is displayed.

AN-3220

A scope capture of the Concentrator Transmissions is shown:

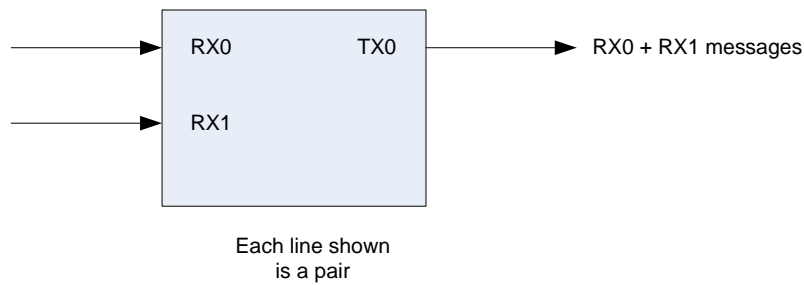


A block diagram shows the paired connections for this demo.



AN-3220

A simplified implementation only requires the two RX0 and RX1 channel inputs, and one TX0 channel output for this concentrator without all the looped-back connections.



Demo 5 uses a single descriptor while Demo 6 used just two descriptors leaving the available descriptor table memory largely unused; this unused descriptor space can be used for other messaging tasks.

TX0 One-Time transmission Demo. Press '7' then from the sub-menu Press '1'.

This demo uses the same exact demo code function as Demo 3, except after a 10ms delay the TX0 Control Register is written again to clear the RUN bit. This demonstrates One-Time transmission of all programmed descriptor messages. After all the messages are sequenced and transmitted, the scheduler stops. View the transmitted messages on a scope on TX0 outputs or make external connections to any receivers and use the '8' and '9' commands to enable reception and display the messages on the console similar to previous demos.

AN-3220

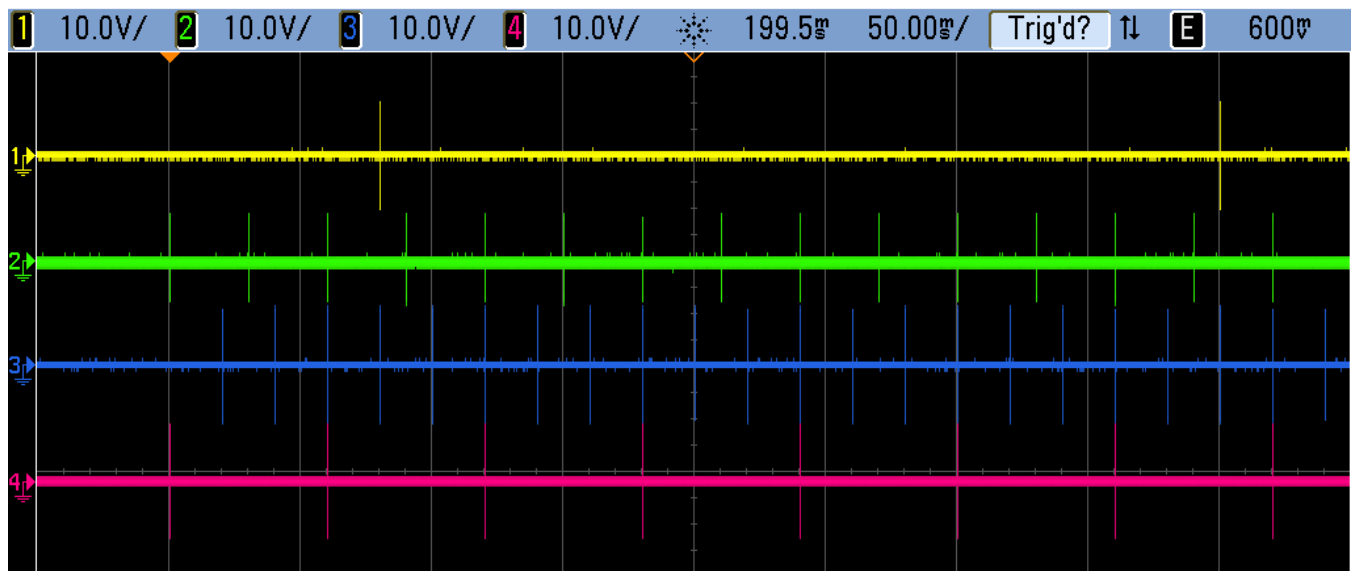
TX0-TX3 Scheduler Packet-Timer Demo. Press '7' then from the sub-menu Press '2'.

The Packet-Timer demo programs the descriptor tables for TX0, TX1, TX2 and TX3 to implement the examples 1 -4 listed in the Packet Timer section of the HI-3220 data sheet. Each example uses different values for PTP (byte3) and PTO (byte4) of the descriptor bytes to demonstrate the flexible timing options of the Transmit Scheduler. In order to duplicate this scope plot, trigger positive on the RUN signal on TP2 using the oscilloscope external trigger input and view TX0, TX1, TX2 and TX3 on a four channel oscilloscope.

Press 7, then Press 2.

Press 'r' to start the scheduler which sets the RUN pin high.

TX0=Yellow, TX1=Green, TX2=Blue, TX3=Red.



Demo P: Both lower case 'p' and upper case 'P' are used in this demo.

Using the HI-3220 with a preprogrammed EEPROM allows auto-initialization and autonomous operation without MCU or FPGA control. Once the configuration data is stored in the EEPROM, the HI-3220 will automatically load all register and selected memory areas shown shaded in the memory map in the data sheet from the EEPROM into the HI-3220 at RESET if the Mode [2:0] inputs are configured for Mode 1 or Mode 3 according to the flow chart on page 27 of the data sheet.

The user must configure Control Registers, Descriptor Tables etc. in the device to achieve the desired functionality before programming the EEPROM with the 'P' command. The 'P' command simply copies the HI-3220 data and register space into the EEPROM.

AN-3220

For this demo, the function `autoInitEEPROMExample()` is called first; this function configures all sixteen Receivers and eight Transmitters for high speed operation in LOOPBACK mode. This configuration is programmed in the EEPROM for this demo.

Demo steps:

1. Set the three Mode pins low, Mode0, Mode1 and Mode2 by placing the jumper shunts in the down position on J7, J8 and J9. Note: This is only required if the EEPROM was never programmed before.
2. Press 'p' to execute the `autoInitEEPROMExample()` function. To test this demo configuration, follow the steps below. This exercise is run to verify that the configuration operates as desired before programming the EEPROM.

- a. Press 'r' to set RUN high.
- b. Press 'k' to transmit a message on TX2 and TX3.
- c. Press '9' to read the received messages. Due to LOOPBACK, each Transmitter is duplicated internally routed to two Receivers.

```
msgs: 1
RX 4:0  02 02 34 56
msgs: 1
RX 5:0  02 02 34 56
msgs: 1
RX 6:0  03 03 78 9A
msgs: 1
RX 7:0  03 03 78 9A
```

3. Press 'P' to program this configuration into the EEPROM. The red ERROR LED flashes briefly.
4. Change the Mode0-2 jumpers for Mode1 or Mode3 and power-cycle the board Off/On or Press RESET. The HI-3220 should auto-initialize from EEPROM.
5. Repeat the sequence steps a-c above to see the demo work the same as before but this time the `autoInitEEPROMExample()` function call is not needed because the configuration was copied into the HI-3220 from the EEPROM by the auto-initialization sequence.

Auto-initialization from EEPROM is an optional feature; an EEPROM is not needed if this feature is not used.

Demo x: Transmits a custom message.

Press 'x' to transmit a custom message on any TX0-TX7 transmitter channel. This utility prompts for a channel number (0-7), message count (0-32), label value (byte), and the Transmit Control Register value then transmits the message(s).

Console Menu Utilities

A SPI peripheral presents some difficulty during development. Traditional bus-interfaced debug environments use memory windows to display memory-mapped registers or memory contents. This capability is not available using SPI, so various SPI utility functions are provided in the Holt demo for writing and reading registers and memory space. These utilities are essential for debug during development, but require a UART/serial port connected to a PC running terminal emulation such as HyperTerminal or TeraTerm. Some of the provided functions write or display a single address location; others access a range of addresses. **All address and data values are hex by default.**

Command 'w':

This command displays the 256 byte region beginning at location 0x8000. This is useful because 0x8000 is the start address for the system registers. After power-up or RESET, nearly all memory contains zeros. Notice the '2' in the upper nibble at address 0x8002 shows the READY bit in the MSR is set high.

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:8000 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 FF
MAP:8010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8040 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00 00 00
MAP:8050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:8070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00
MAP:8080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=====
Keys: 'W' Watch  '<' Up  '>' Down  'R' Refresh  'A' Address
0x8000-0x808F  End MAP:8090

```

Pressing the < key subtracts 256 from the start address and pressing > adds 256 to the start address.

Command 'a':

This prompts for a 4-digit hex address and displays that 256 byte region.

```

a
Enter 16-bit hex address: 4000
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:4000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:4090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:40A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:40B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:40C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:40D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:40E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:40F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=====
Keys: 'W' Watch  '<' Up  '>' Down  'R' Refresh  'A' Address
0x4000-0x40FF  End MAP:4100

```

Command 'c':

Sometimes it is useful to clear all device memory from the console instead of using reset or power cycling the board. This guarantees fresh initialization values and device memory is cleared to zeros making it easier to identify updated values later. Press 'c' to clear all memory.

Command 'd':

This command prompts for a channel value and expects 0-F to be entered. The corresponding RX RAM message buffer is displayed for all 256 labels. The utility continuously prompts for another value to repeat the process. Press 'q' to quit the utility.

Command 'L':

This command demonstrates two useful functions that set or clear ARINC Labels in the Received Data Enable Look-Up table; refer to HI-3220 data sheet page 20. Functions `setLabel(chx, label)` and `clearLabel(chx, label)` can be called by the user in their own C projects. The input parameters for these functions are Receiver channel number (0-15) and label (0-255 dec). Setting the appropriate bit manually in the table is otherwise a tedious process. A Data Enable Look-Up table diagram is provided in the data sheet, showing how the label enable bits are arranged in the table memory.

Press 'L' to set the labels called out in the demo.

The following sequence takes place.

1. Channel CH0 labels are set: 0, 16, 127, 128, 254.
2. Channel CH7 labels are set: 1, 17, 128, 255.
3. Channel CH15 label 255 is set.
4. The table memory is displayed on the console.
5. Channel CH15 label is cleared.
6. The table memory is displayed again on the console showing revealing CH15 label 255 is now cleared.

Results

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:6800 01 00 01 00 00 00 00 00 00 00 00 00 00 00 80  RX0 Labels 0, 16, 128 set
MAP:6810 01 00 00 00 00 00 00 00 00 00 00 00 00 00 40  RX0 Labels 128, 254 set
MAP:6820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68E0 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00  RX7 Labels 1, 17 set
    
```

AN-3220

```
MAP:68F0 01 00 00 00 00 00 00 00 00 00 00 00 00 00 80 RX7 Labels 128, 255 set
MAP:6900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 ←RX15 label 255 set
```

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
MAP:6800 01 00 01 00 00 00 00 00 00 00 00 00 00 00 00 80
MAP:6810 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40
MAP:6820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68E0 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:68F0 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80
MAP:6900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:6990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MAP:69F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ←- CH15 label 255 cleared
```

Optionally use the 'a' or 'e' command to display this region of device memory.

Command 'm': (modify register/memory value)

This command is used to write a byte to any 16-bit address.

```
> m
```

```
Enter 16-bit hex address: 69ff
```

```
Enter hex byte : 80 readback: 80 (This action just re enabled CH15 label 255 again from the
previous demo.
```

There are a few more utility commands that the user can explore and might find useful.

Application considerations

ARINC 429 Reception by RAM vs. Reception by FIFO (and host keeping pace with data)

There are two methods to read ARINC 429 messages. Reception by RAM access uses a single 4-byte buffer location mapped in 16K device memory according to channel number (0-15) and Label value. See the data sheet on page 18 for the memory organization. Reception by FIFO access uses a 64 message FIFO for each Receiver channel (0-15).

The preferred method depends on user requirements, speed of host controller, software overhead and ARINC 429 traffic. When using Reception by RAM access, the host must read a message on the same channel and same Label before the next message arrives, or next message data will overwrite the previous data. If the next message is on a different channel or different label, no overwrite occurs because that channel or label maps into a different 4-byte buffer location. An INT interrupt could be used to achieve fast response to reading these messages. In high speed, ARINC 429 messages could arrive about every 360µs.

Using FIFO access helps alleviate tight timing constraints by storing up to 64 messages before potential overwrite occurs when the host fails to read messages fast enough to keep up with incoming traffic. RX channel interrupts and a FIFO Threshold register provide flexibility for designing a solution that processes messages fast enough to meet application requirements. Because of the FIFO's ability to store up to 64 messages before a possible overwrite (or data loss) occurs, a polling strategy is often used instead of an interrupt.

The easiest way to configure the device to receive ARINC 429 messages is using FIFO Reception method. The easiest way to configure and transmit ARINC 429 data is to use the MCU Direct Transmit method.

Improved Host Receiver FIFO reading efficiency with the HI-322x family

A new feature in the HI-3220 family is the ability to read the current message count of either the Transmit FIFOs or the Receiver FIFOs. This allows the host to read a channel's FIFO count and pass that count to a Read_FIFO16 function that reads all the messages in one shot. In addition a new C driver reads 16-bits at a time. This feature is used in the demos in several places, but the two essential lines of code are.

```
msgCnt = ReadRegister(FCVOR, 0);           Fetch message count of Receiver RX0 FIFO
Read_FIFO16(ch0, bufferRX, msgCnt);       Reads all FIFO messages into the buffer
```

EEPROM auto-initialization feature

The optional EEPROM auto-initialization feature allows the device to automatically copy register, Transmit Scheduler tables, Interrupt Map and FIFO Enable Map values from a preprogrammed EEPROM into device memory space after power-up or reset. This feature is useful if it's necessary to initialize and run the device before a host MCU or FPGA can perform initialization, or if the application is an autonomous Data Concentrator or Repeater that operates autonomously, without a host MCU or FPGA. In most cases, if a host is available, "initialize from EEPROM" is not typically needed.

When not using an EEPROM, the Mode0-2 pins must be tied either to VDD or GND through 1K to 3.3K resistors, depending on the desired power-up mode. These pins do not have integrated pull-up or pull-down resistors and 3-bit mode must be either 0, 1, 2 or 5 as indicated in Figure 1 of the data sheet.

HI-3220 SPI basics for troubleshooting

This is a brief tutorial on host SPI communication with the HI-3220. Four simple SPI sequences are generated when the SW2 push-button switch is pressed, as explained below. The logic analyzer plots show the waveforms for each of these. This information may be helpful when troubleshooting the Master SPI interface.

The four separate SPI sequences that produce these waveforms are shown below.

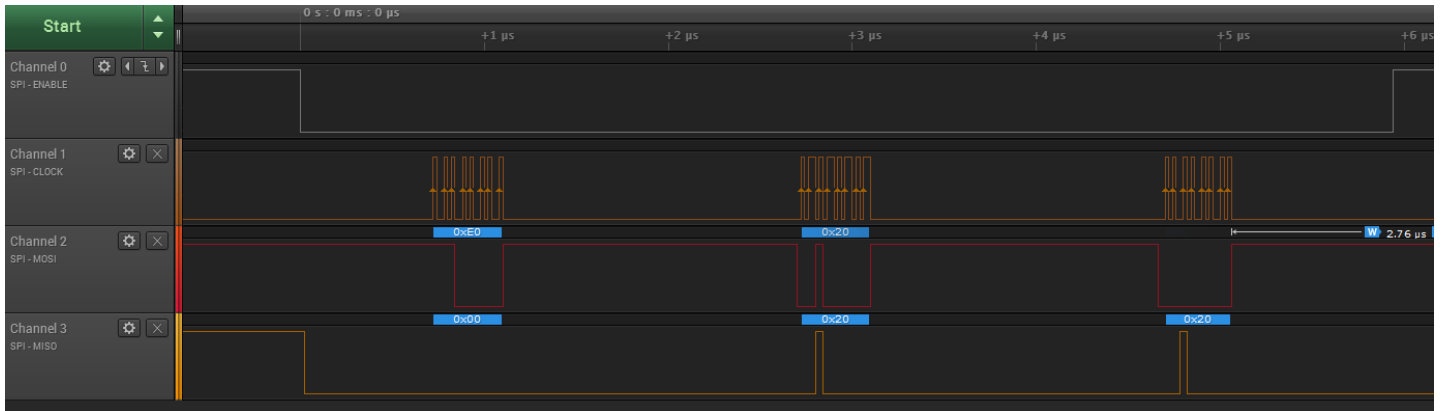
```
retVal = ReadRegister(0x02, 0);      // read MSR register 0x8002
Write_MAP (0x8000,0);                // write 0x8000 address (MCR) to the MAP pointer
Write_MAP_Byte (0xC0,0);             // write 0xC0 to MCR - MAP points to MCR register
retVal2 = Read_MAP_Byte(0);          // read byte at existing MAP pointer
```

The values of retVal and retVal2 are displayed on the console when SW2 is pressed.

```
Simple SPI example
MSR = 20
MCR = C0
```

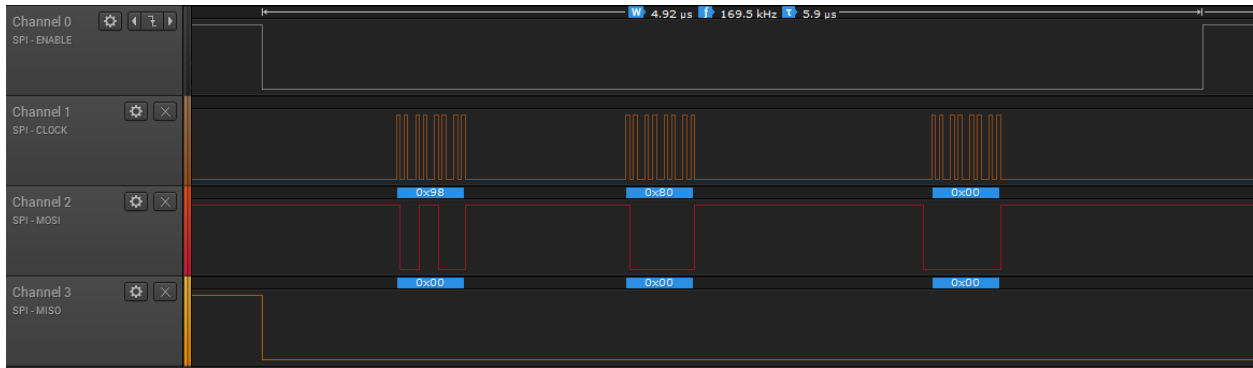
The simplest SPI operation that can be used to test the interface is the Read Register Op Code to read the value of the Master Status Register(MSR) which should contain 0x20 with bit-5 READY set high. When the HI-3220 becomes READY after RESET, the READY output pin is set high and bit-5 should also be high in the MSR if when read by SPI. The Register Read two byte Op Code 0xE0 + 0x20 consists of an Op Code byte followed by another byte for the lower portion of the 16-bit address. The AAA-AAAA field is split between the first Op Code byte and the second Op Code byte. This Op Code automatically adds 0x80 for the upper address byte to the lower address byte provided by the AAA-AAAA field so the address is 0x8002. The Master Status register location is 0x8002 so that is the register that is returned with this Op Code. The SPI sequence is shown below and the user should verify the Master SPI signaling is similar to this waveform. Notice SO data is clocked out on the negative edge of SPI clock and the Master should clock data on the rising edge. This is SPI mode 0.

AN-3220

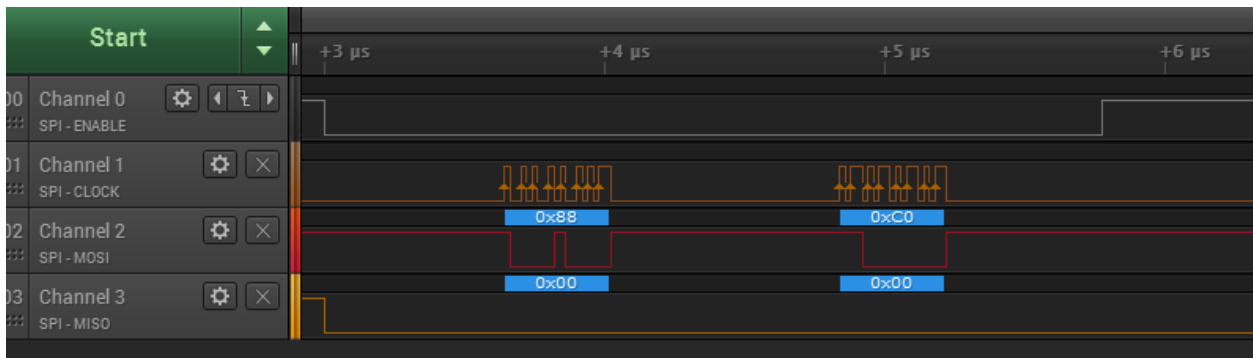


Read the MSR (0x8002)

If the Read Register Op Code is working properly, the user can proceed to the next steps to verify the following three SPI sequences which determines whether SPI writes are working. The MAP (memory address pointer) register is used to write and read back a register (or memory) location. First an address is written to the 16-bit MAP pointer then a byte value is written to this location using an Op Code that uses the MAP pointer value for the write address. The value is then read back, also using the same MAP pointer address. This should return the original byte value that was originally written. See the following three SPI sequences for these examples.

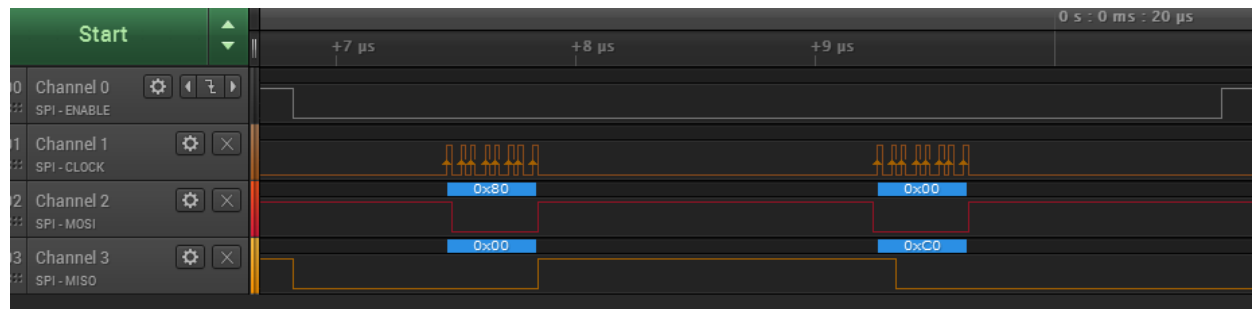


Write address 0x8000 to the MAP pointer



Write value 0xC0 to MAP address

AN-3220



Read value at MAP address (should be 0xC0)

Final Tips:

1. The READY output pin must be high in order to write to any Control Register or other memory space. READY is asserted high after a qualifying negative reset pulse on the /MRST input.
2. When loading Transmitter Descriptor tables, always load an EOS (0x00) byte after the last loaded descriptor to terminate the descriptors unless the table is fully loaded.
3. The RUN pin must be high in order for the Receivers or Transmitters to operate.
4. When using the EEPROM to auto-initialize, it must have been previously programmed. If the HI-322x powers-up in Mode 1 or Mode 3 with an unprogrammed EEPROM, a fault will occur and the red LED goes On.
5. When initializing RX Receiver channels for FIFO message reception remember to also initialize the Received FIFO Enable Look-Up Table so the desired labels are enabled. Logic-1 enables a label and 0 disables the label. The device powers up with all the label entries cleared to 0x00 so by default all labels are disabled. If a label is disabled, no message reception on that channel will occur. The demos typically set all the label table memory to 0xFF.

Getting Started with the Holt API demo software project and installing IAR Systems *Embedded Workbench for ARM* Compiler

1. Installed IAR Systems *Embedded Workbench for ARM* (EWARM) compiler is required BEFORE adding the Holt demo project, so all Atmel board library files and the demo project folder are created in the proper locations. Follow the “*Holt HI-3220 Demo Project Installation Guide*” found in the Project folder on the Holt CD-ROM. Before proceeding to the next steps IAR must be installed and the Holt project folder must be in the proper folder location, according to that guide. **Instructions beyond this point assume you have completed the above installation tasks.** The Holt demo project requires IAR EWARM version 7.1 or greater. If using IAR 8.x versions, another technical note provided additional guidance.
2. Launch IAR *Embedded Workbench* from the Windows Start menu. A blank screen should appear. Open the Holt HI-3220 Demo Project from the IAR File pull-down menu, click on File/Open/Workspace and navigate to the project folder location and select “HI-3220 Demo.eww” and click the Open button.
3. Debug requires an interface between the computer running IAR Embedded Workbench® and the HI-3220 Application Development Kit. Connect the small end of the provided USB cable to the evaluation board USB connector marked DEBUG. Connect the other end of the USB cable to a free computer USB port. The IAR C-SPY Debugger for ARM includes drivers for numerous target system interfaces, including built in “J-link On Board”.

The first time the evaluation board USB cable is connected to the computer, the Windows “Found New Hardware” message should appear for the J-Link device. After several seconds, Windows should load the appropriate driver and advise, “Your hardware is ready for use”. If Windows fails to find the J-Link driver, direct it to look in the Drivers directory the IAR Embedded Workbench® installation CD.

If difficulties arise when initiating a debug session at step 5, click **Project** then **Options**. In the window that opens, under **Category = Debugger** highlight **J-Link/J-Trace**. Click the tab labeled **Connection**, then verify Communications = USB and Interface = SWD.

3. Open IAR Embedded Workbench®. Click **File**, then **Open Workspace**, then navigate to the project subdirectory created in step 4. Select the project file with .eww extension, then click **Open**. (The next time Embedded Workbench® opens, this project will appear in the Recent Workspaces list when **File** is clicked.)

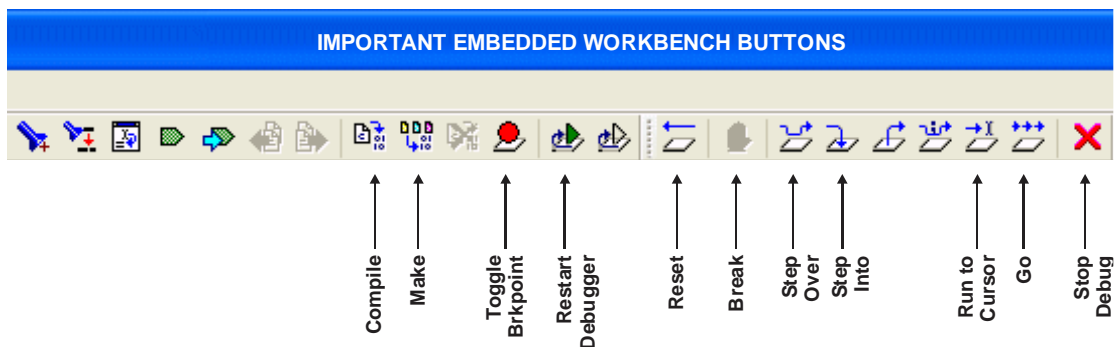
4. If problems occur with IAR installation or with using the IAR debugger, two Holt technical notes are provided to help resolve these issues included on the Holt CD ROM.

5. The demo project only uses unsigned integer variables. Optionally turn off the nuisance compiler message that occurs when a variable's most significant bit toggles. The message looks like this:
Remark[Pe068]: integer conversion resulted in a change of sign
 To disable this diagnostic message, click **Project** then click **Options**
 Category = C/C++ Compiler
 Tab = Diagnostics
 Suppress these diagnostics: add "Pe068" to list

6. RAM based projects are not supported due to the limited amount of RAM on the MCU. By design the Cortex™-M3 runs slower in RAM than in Flash so there is little need for a RAM based project.

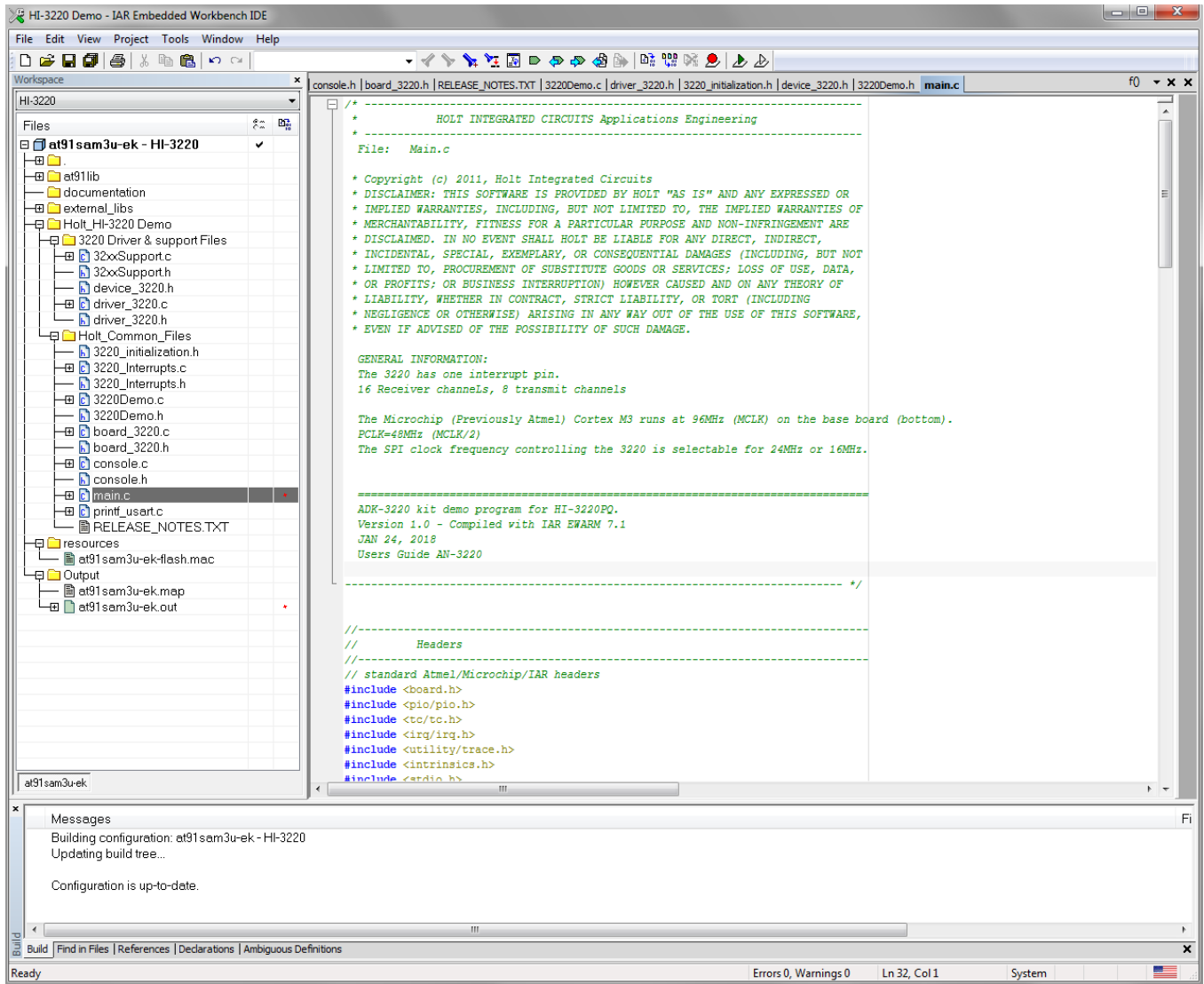
Compile the project by clicking the **Make** button. See following illustration. If the Build messages window in IAR Embedded Workbench® indicates no errors or warnings, you can continue. If errors occurred, correct them and recompile the program.

11. Initiate a debug session by clicking the **Restart Debugger** button. This downloads the compiled program into the MCU and readies the board for program execution. Click **Go** to start execution. Click **Break** (normally displayed during execution as a red upheld hand) to stop execution.



AN-3220

The IAR IDE screen with the Holt HI-3220 project loaded. Resized the IDE screens may be necessary.



Project File List with Selected Descriptions

HEADER FILES WITHOUT CORRESPONDING C FILES

device_3220.h

Contains all macros for the HI-3220 including define statements for registers, control register bits, Op Codes and selected table start addresses.

3220_initialization.h

Definitions for a few configuration settings including the INTERRUPT_MESG_ENA macro used to disable or enable interrupts in the demo.

HI-322x Low-Level SPI drivers and other 3220 specific support functions

device_3220.h (same as above)

Contains all macros for the HI-3220 including define statements for registers, control register bits, Op Codes and selected table start addresses.

driver_3220.c/driver_3220.h

Host Cortex M3 SPI low-level SPI drivers and messaging functions.

32xxSupport.c/32xxSupport.h

Host helper functions for configuring, table initialization and transmitting and receiving ARINC messages.

Other main C FILES WITH CORRESPONDING HEADER FILES

Most of the function names are self-explanatory.

main.c

The primary program entry and main loop in main().

Board_3220.c

Contains macros for SPI initialization, LEDs and various I/Os. Timer1 tick lms initialization supporting delay functions and host timer interrupt handler. Board related to the host MCU.

3220_Interrupts.c

Initialization and IRQ handler for message interrupts on the 3220 INT pin. Functions to set or clear interrupt label enable table memory.

3220Demo.c (these demos are called mostly from the console)

All demos are contained here. See the header file for a complete listing.

console.c

Console functions used by all demos.

Show_menu(); displays the main menu

chk_key_input(); Key entry detect and demo execution handler.

Application Development Kit Notes

The HI-3220 was designed for compatibility with microcontrollers or FPGAs having a Serial Peripheral Interface (SPI). Device RAM and register locations are written or read with the help of 8-bit SPI commands. Most read or write operations use a Memory Address Pointer (MAP) to designate the address of the next location accessed. To speed up multi-word transfers, the Memory Address Pointer automatically increments to the next address after each word read or write is performed.

The HI-3220 data transfer speed depends on the SPI clock frequency provided by the MCU SPI interface. The ARM Cortex M3 MCU master clock frequency is 96MHz, using an internal PLL. MCU peripherals divide this clock by two for a 48MHz clock source. The SPI block divides this by three for a 16MHz SPI clock or divides by 2 for 24MHz SPI clock. The HI-3220 has a maximum SPI clock frequency of 40MHz.

The demo board includes a fully featured high-end USB debugger port. No additional debug tools are required to rebuild the demo project and re-flash the ARM Cortex M3 processor.

HI-3220 use with an external MCU or FPGA

Using the HI-3220 daughter card with an external MCU or FPGA is made easy because the 50MHz oscillator is contained on this card and the essential interface signals are available on J11 header connector. See the schematic and the header pin out description in a previous section of this document for those details.

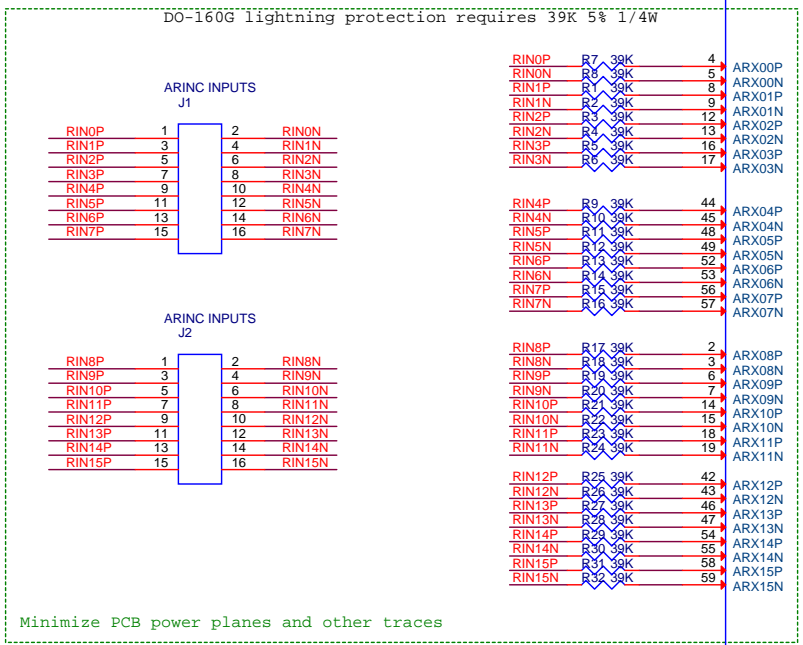
Summary

The HI-3220 is a powerful ARINC 429 multi-channel Transmitter and Receiver solution. Its high-speed 4-wire SPI interface makes it easy to control. The maximum 40MHz SPI rate and new features in the HI-3220 increases the host communication efficiency. The HI-3220 can auto-initialize from EPPROM and operate without a FPGA or MCU in some applications. The demo program demonstrates nearly all aspects of the device and the low level C drivers make it easy to port to other platforms. A rich set of menu SPI access utilities allow viewing/modifying device memory on the console.

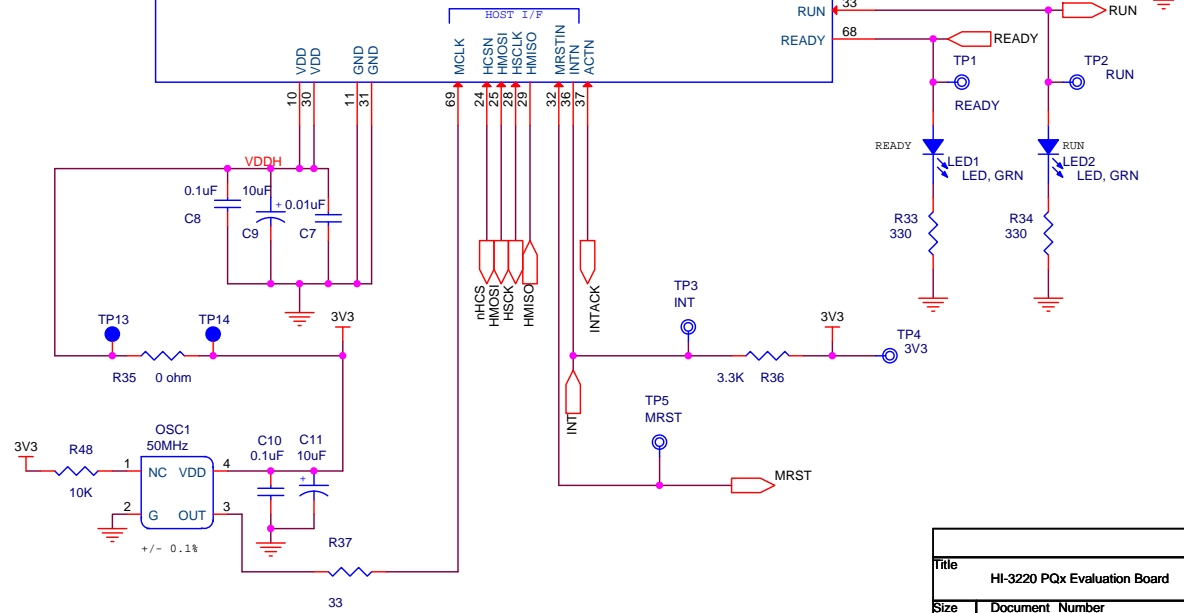
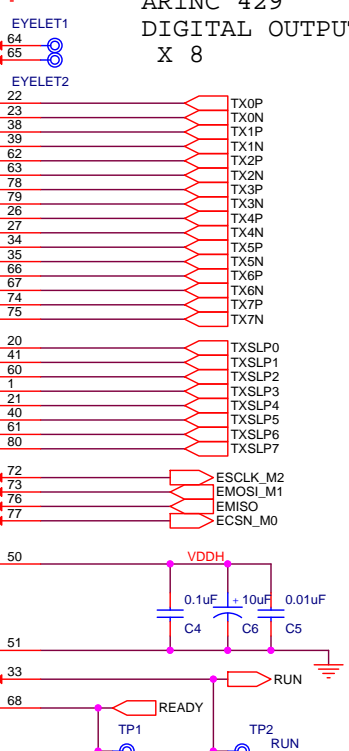
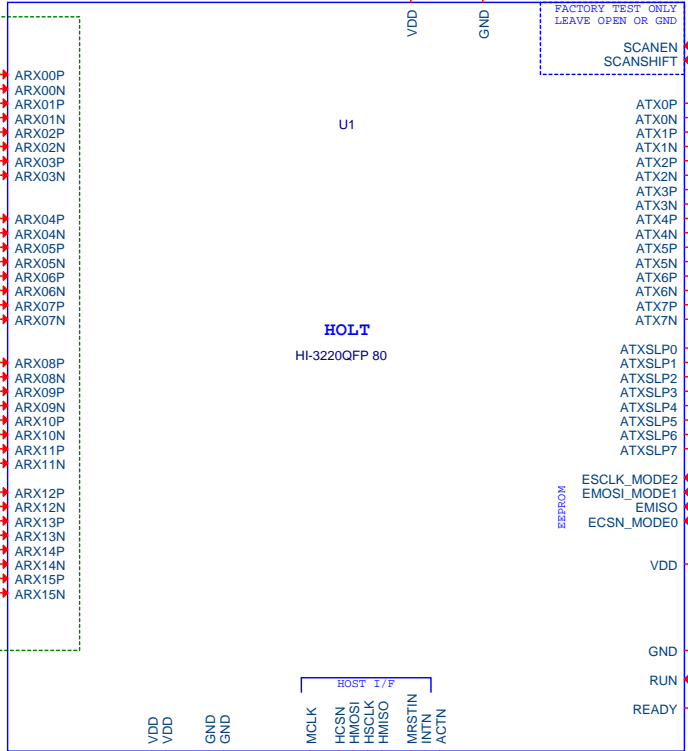
Item	Qty	Description	Reference	Digikey P/N	Mfg P/N
1	1	PCB, Bare, Evaluation Board	N/A		JetTech # 42174
2	5	Capacitor Cer 0.1uF 25V 10% X7R 0805	C1,C4,C8,C10,C48	478-3755-1-ND	AVX 08053C104KAT2A
3	3	Capacitor Cer 0.01uF 100V 10% X7R 0805	C2,C5,C7	478-1358-1-ND	AVX 08051C103KAT2A
4	16	Capacitor Cer 4.7uF 16V 10% X7R 0805	C14-C17,C22-C25,C32-C35,C40-C43	587-3312-1-ND	Taiyo EMK212AB7475KGHT
5	4	Capacitor Tant 10uF 10V 10% 1206	C3,C6,C9,C11	399-3684-1-ND	Kemet T491A106K010AT
6	24	Capacitor Cer 10uF 16V 20% X7R 0805	C12-C13,C18-C21,C26-C31,C36-C39,C44-C47,C49-C52	587-3319-1-ND	Taiyo EMK212BB7106MG-T
7	1	Resistor 0, 1/8W 5% 0805 SMD	R35	P0.0ACT-ND	Panasonic ERJ-6GEY0R00V
8	1	Resistor 33, 1/8W 5% 0805 SMD	R37	P33ACT-ND	Panasonic ERJ-6GEYJ330V
9	7	Resistor 330, 1/8W 5% 0805 SMD	R33-R34,R38-R42	P330ACT-ND	Panasonic ERJ-6GEYJ331V
10	3	Resistor 1K, 1/8W 5% 0805 SMD	R43-R45	P1.0KACT-ND	Panasonic ERJ-6GEYJ102V
11	1	Resistor 3.3K, 1/8W 5% 0805 SMD	R36	P3.3KACT-ND	Panasonic ERJ-6GEYJ332V
12	3	Resistor 10K, 1/8W 5% 0805 SMD	R46-R48	P10KACT-ND	Panasonic ERJ-6GEYJ103V
13	32	Resistor 39K, 1/4W 0.5% 0805 SMD	R1-R32	P21199CT-ND	Panasonic ERJ-PB6D3902V
14	3	Test Point, Black Insulator, 0.062"	GND (TP10-TP12)	36-5011-ND	Keystone 5011
15	1	Test Point, Red Insulator, 0.062"	3V3 (TP4)	36-5010-ND	Keystone 5012
16	1	Test Point, White Insulator, 0.062"	RUN (TP2)	36-5012-ND	Keystone 5012
17	1	Test Point, Orange Insulator, 0.062"	RDY (TP1)	36-5013-ND	Keystone 5013
18	6	LED Green 0805 SMD	LED1,2,3,4,5,7	160-1423-1-ND	LiteOn LTST-C171GKT
19	1	LED Red 0805 SMD	LED6	160-1422-1-ND	LiteOn LTST-C171EKT
20	3	Header, Male 1x3, .1" Pitch	J7-J9	S1012E-03-ND	Sullins PEC03SAAN
21	1	Header, Male 1x5, .1" Pitch	J5	S1012E-05-ND	Sullins PEC05SAAN
22	3	Header, Male 1x8, .1" Pitch	J4,J6,J8	S1012E-08-ND	Sullins PEC08SAAN
23	1	Header, Male 1x12, .1" Pitch	J11	S1012E-12-ND	Sullins PEC12SAAN
24	2	Header, Male 2x3, .1" Pitch	J-5A	S2012EC-03-ND	Sullins PREC003DAAN-RC
25	2	Header, Male 2x5, .1" Pitch	J_5B	S2012EC-05-ND	Sullins PREC005DAAN-RC
26	2	Header, Male 2x20, .1" Pitch	J_3,J_4	S2012EC-20-ND	Sullins PREC020DAAN-RC
27	1	Switch Tape Seal 4 Pos SMD	SW1	CT2194MST-ND	CTS 219-4MST
28	1	IC EEPROM 512KBIT 20Mhz 8SOIC	U10	CAT25512VI-GT3OSCT-ND	On Semi CAT25512VI-GT3
29	1	Oscillator XO 50.0Mhz HCMOS SMD	OSC1	535-9330-1-ND	Abracon ASV-50.000MHZ-EJ-T
30	8	HI-8597PS	U2-U9	Holt Inc.	HI-8597PS 16-ESOIC
31	1	HI-3220PQ	U10	Holt Inc.	HI-3220 80-QFP

ARINC 429 RECEIVERS X 16

ARINC 429 DIGITAL OUTPUTS X 8

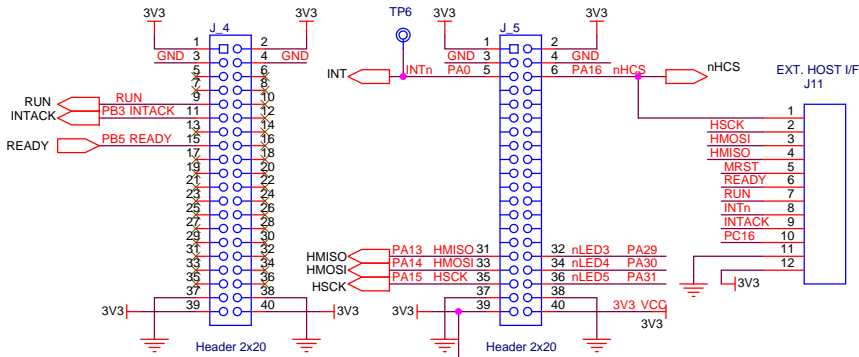


Minimize PCB power planes and other traces



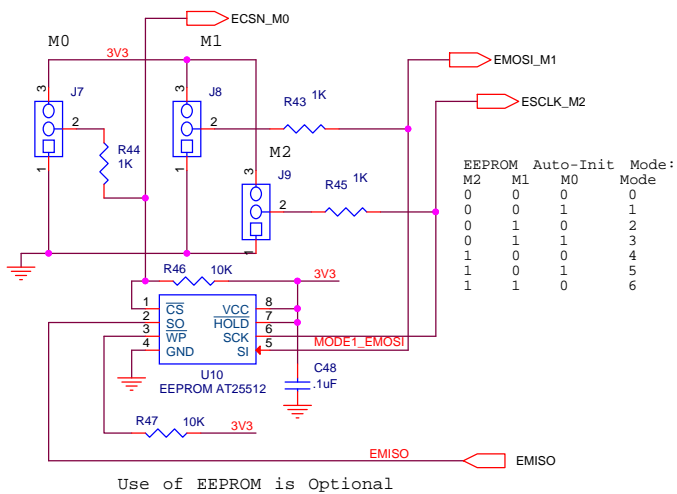
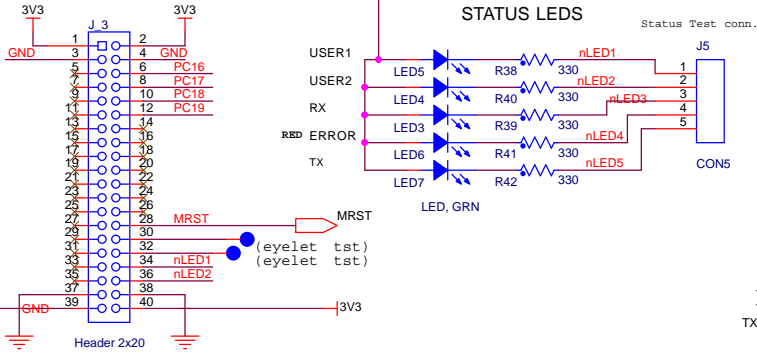
Title		
HI-3220 PQx Evaluation Board		
Size	Document Number	Rev
Custom		NEW
Date:	Tuesday, December 19, 2017	Sheet 1 of 3

HOST INTERFACE

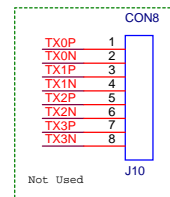
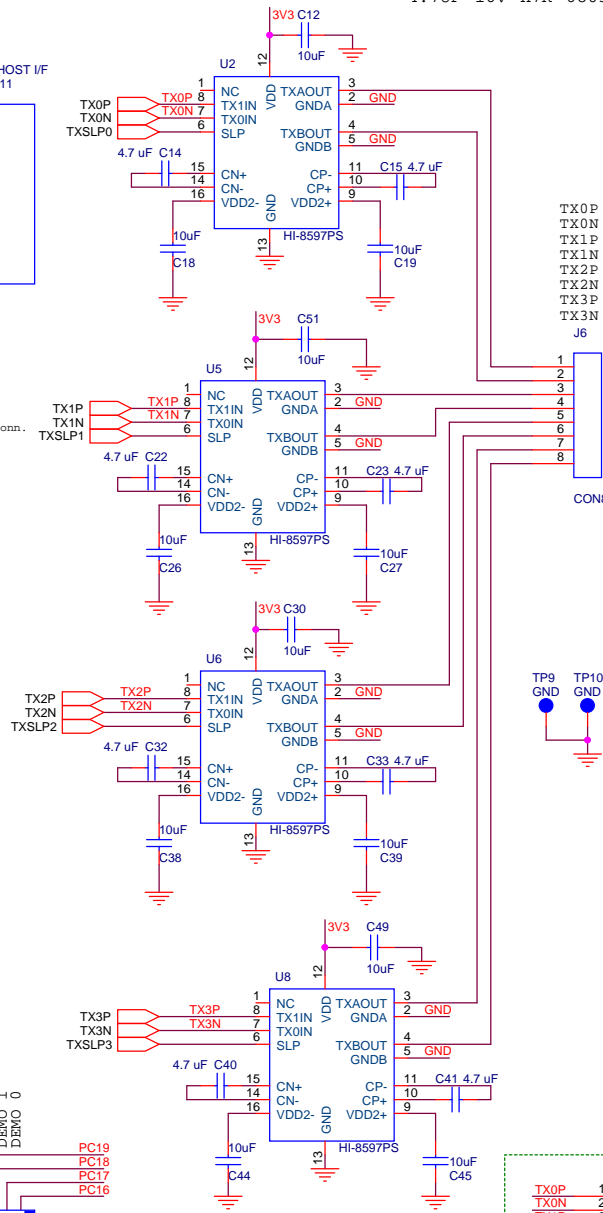
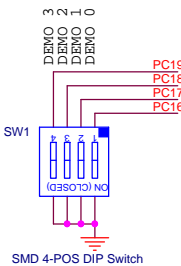


HI-8597 Low ESR ceramic ML capacitors:
 10uF 16V X7R 0805 587-3319-1-ND
 4.7uF 16V X7R 0805 587-3312-1-ND

STATUS LEADS



DEMO SWITCHES



HOLT INTEGRATED CIRCUITS INC.		
Title	HI-3220PQx Evaluation Board	
Size	Document Number	Rev
Custom	<Doc>	NEW
Date:	Tuesday, December 19, 2017	Sheet 2 of 3

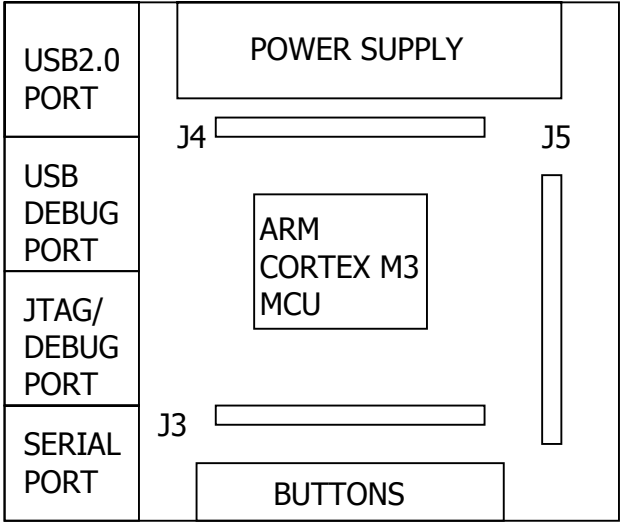
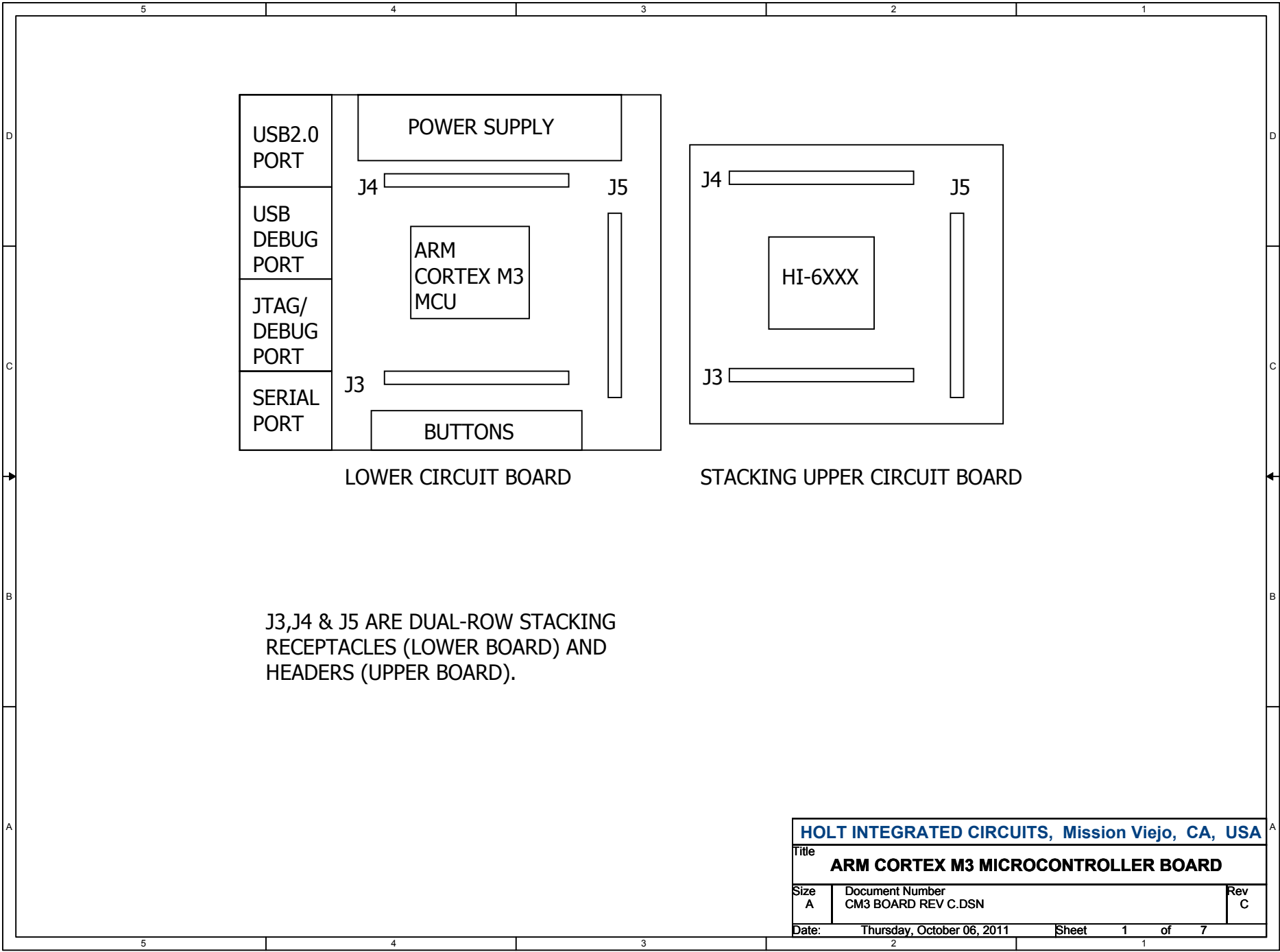
Use of EEPROM is Optional

SMD 4-POS DIP Switch

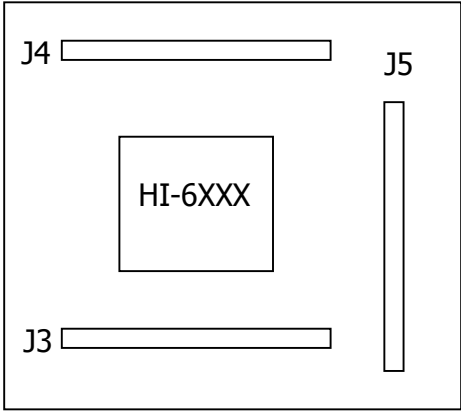
Not Used

Bill of Materials		ARM Cortex M3 MCU Board			Revised: 9 Sept 2011
Item	Qty	Description	Reference	DigiKey	Mfr P/N
1	1	PCB, Bare, Evaluation Board, revision B or C	N/A	-----	
2	1	Ferrite Bead, 220 Ohm @ 100MHz 300mA DC 0805	FB1	732-1602-1-ND	Wurth 742792034
3	2	Capacitor, Ceramic 10nF 10% 50V X7R 0805	C1,C42	399-1158-1-ND	Kemet C0805C103K5RACTU
4	2	Capacitor, Ceramic 10pF 10% NP0 C0G 0V 0805	C23,C34	478-3731-1-ND	AVX 080551A100KAT2A
5	4	Capacitor, Ceramic 20pF 5% NP0 C0G 0V 0805	C14,C21,C25, C27	478-3735-1-ND	AVX 080551A200JAT2A
6	28	Capacitor, Ceramic 100nF 20% 50V Z5U 0805	C2,C4,C6-C11, C13,C15-C19, C22,C24,C26, C28,C29,C33, C35-C40,C45-46	399-1176-1-ND	Kemet C0805C104M5UACTU
7	4	Capacitor, Tantalum 4.7uF 10% 10V Low ESR SMD 1206	C5,C20,C31, C32	478-2391-11-ND	AVX TPSA475K010R1400
8	4	Capacitor, Tantalum 10uF 10% 10V Low ESR SMD 1206	C3,C12,C30,C41	478-3317-1-ND	AVX TPSA106K010R1800
9	1	Capacitor 22uF 10% 6.3V Tantalum Low ESR SMD C	C43	495-1504-1-ND	Kemet B45197A1226K309
10		Capacitor 100uF 10% 6.3V Tantalum Low ESR SMD C	C44	495-1509-1-ND	Kemet B45197A1107K309
11	1	Header, Male Shrouded 2x10 0.1" Pitch	J1	MHB20K-ND	3M 2520-6002UB
12	1	Connector, Receptacle USB Mini B Rt-Angle PCB Mount	J2	H2959CT-ND	Hirose UX60-MB-5ST
13	1	Connector DB9F, Right-Angle PCB Short Body, Board Lock	J6	182-109FE-ND	NorComp 182-009-213R-561
14	1	Jack, DC Power, 2.5mm ID x 2.1mm pin	J7	CP-102AH-ND	CUI PJ-102AH
15	2	Receptacle, Female 2x20 0.1" Pitch, 8.5mm Height, 3.2mm Solder Tails	J3,J4	S6104-ND	Sullins PPTC202LFBN-RC
16	1	Receptacle, Female 2x4 0.1" Pitch, 8.5mm Height, 3.2mm Solder Tails	J5A (J5 lower end, close to Bus B)	S7072-ND	Sullins PPTC042LFBN-RC
17	1	Receptacle, Female 2x5 0.1" Pitch, 8.5mm Height, 3.2mm Solder Tails	J5B (J5 upper end, close to Bus A)	S6105-ND	Sullins PPTC052LFBN-RC
18	1	Solder Jumper	JP1	SOLDER CLOSED	
19	2	Inductor, 10uH, 100mA 0805	L1,L2	490-4029-1-ND	Murata LQM21FN100M70L
20	1	LED Green 0805	LED1	160-1179-2-ND	LiteOn LTST-C170GKT
21	0	Resistor, Prov 1/8W 0805	R1	DO NOT STUFF	
22	5	Resistor, 0 ohm 1/8W 0805	R12,R13,R22, R23,R29	311-0ARCT-ND	Panasonic ERJ-6GEY0R00V
23	2	Resistor, 1.0 5% 1/8W 0805	R7,R8	P1.0ACT-ND	Panasonic ERJ-6GEYJ1R0V
24	2	Resistor, 39 5% 1/8W 0805	R4,R5	P39ACT-ND	Panasonic ERJ-6GEYJ390V
25	1	Resistor, 150 5% 1/8W 0805	R17	P150ACT-ND	Panasonic ERJ-6GEYJ151V
26	1	Resistor, 4.7K 5% 1/8W 0805	R3	P4.7KACT-ND	Panasonic ERJ-6GEYJ472V
27	1	Resistor, 6.8K 5% 1/8W 0805	R6	P6.8KACT-ND	Panasonic ERJ-6GEYJ682V
28	1	Resistor, 47K 5% 1/8W 0805	R18	P47KACT-ND	Panasonic ERJ-6GEYJ473V
29	1	Resistor, 68K 5% 1/8W 0805	R19	P68KACT-ND	Panasonic ERJ-6GEYJ683V
30	10	Resistor,100K 5% 1/8W 0805	R2,R10,R11, R20,R21,R24, R25,R26,R27, R28	P100KACT-ND	Panasonic ERJ-6GEYJ104V
31	3	Pushbutton	SW1,SW2,SW3	P10886SCT-ND	Panasonic EVQ-QWS02W
32	2	Test Point, Black Insulator, 0.062" hole	TP2,TP3	5011K-KD	Keystone 5011

33	1	Test Point, Orange Insulator, 0.062" hole	TP1	5008K-ND	Keystone 5008
34	1	Test Point, Yellow Insulator, 0.062" hole	TP4	5009K-ND	Keystone 5009
35	2	Test Point, Hole / Pad Only	TP5,TP6		
36	1	IC, MCU 32-Bit 256KB Flash, 144-LQFP	U1	ATSAM3U4EA-AU-ND	Atmel ATSAM3U4EA-AU
37	1	IC, ESD Protection Array 3-Channel SOT-5	U2	296-21885-1-ND	Texas Inst TPD3E001DRLR
38	1	IC, RS232 Driver/Receiver 3.0 to 5.5VDC 16-SOIC (3.9mm wide)	U3	296-19752-1-ND	Texas Inst MAX3232EIDR
39	1	IC, Single Inverter 74LVC1G04 SC70-05	U4	296-11600-1-ND	Texas Inst SN74LVC1G04DCKR
40	1	IC Voltage Regulator 3.3V 1A LDO, SOT-223	U5	497-1228-1-ND	ST Micro LD1117AS33TR
41	1	PolyZen 5.6V PPTC protected Zener SMD	U6	ZEN056V130A24LSCT-ND	Tyco ZEN056V130A24LS
42	1	Filter, EMI 35dB 10A 1MHz-1GHz SMD	U7	490-5052-1-ND	Murata BNX022-01L
43	1	IC Voltage Ref 2.5V 1% Micropower SOT-23	VR1	576-1047-1-ND	Micrel LM4040DYM3-2.5
44	1	Crystal 12.00MHz, 50ppm 20pF, HC-49US leaded	Y1	631-1105-ND	Fox FOXSLF/120-20
45	1	Crystal, 32768 Hz 12.5pF cylinder leaded	Y2	535-9033-1-ND	Abrakon AB26TRB-32.768KHZ-T
J-Link On-Board Circuitry...					
46	10	Capacitor, Ceramic 100nF -20% / +80% 25V Y5V 0603	C48-C53, C55-C58	490-1575-1-ND	Murata GRM188F51E104ZA01D
47	1	Capacitor, Ceramic 33pF 5% 50V C0G 0603	C59	490-1415-1-ND	Murata GRM1885C1H330JA01D
48	2	Capacitor, Ceramic 15pF 5% 50V C0G 0603	C60,C61	490-1407-1-ND	Murata GRM1885C1H150JA01D
49	2	Capacitor, Ceramic 10pF 5% 50V C0G 0603	C62,C63	490-1403-1-ND	Murata GRM1885C1H100JA01D
50	1	Capacitor, Ceramic 1nF 20% 50V X7R 0603	C64	490-1495-1-ND	Murata GRM188R71H102MA01D
51	1	Capacitor, Ceramic 10nF 10% 50V X7R 0603	C65	490-1512-1-ND	Murata GRM188R71H103KA01D
52	1	Capacitor, Ceramic 4.7uF -20% / +80% 6.3V 0603	C47	587-1313-1-ND	Taiyo Yuden JMK212F475ZD-T
53	1	Ferrite Bead, 220 Ohm @ 100MHz 300mA DC 0805	FB2	SAME AS FB2 ABOVE	
54	1	Solder Jumper	JP2	LEAVE OPEN	
55	1	Connector, Receptacle USB Mini B Rt-Angle PCB Mount	J8	SAME AS J2 ABOVE	
56	1	LED Green 0805	LED2	SAME AS LED1 ABOVE	
57	1	Resistor, 0 ohm 1/10W 0603	R30	P0.0GCT-ND	Panasonic ERJ-3GEY0R00V
58	1	Resistor, 220 ohm 5% 1/10W 0603	R31	P220GCT-ND	Panasonic ERJ-3GEYJ221V
59	2	Resistor, 1.5K ohm 5% 1/10W 0603	R32,R41	P1.5KGCT-ND	Panasonic ERJ-3GEYJ152V
60	1	Resistor, 47K ohm 5% 1/10W 0603	R33	P47KGCT-ND	Panasonic ERJ-3GEYJ473V
61	1	Resistor, 100 ohm 5% 1/10W 0603	R34,R35,R37 R39	P100GCT-ND	Panasonic ERJ-3GEYJ101V
62	1	Resistor, 27 ohm 5% 1/10W 0603	R36,R38	P27GCT-ND	Panasonic ERJ-3GEYJ270V
63	1	Resistor, 300 ohm 5% 1/10W 0603	R40	P300GCT-ND	Panasonic ERJ-3GEYJ301V
64	1	IC AT91SAM7S64 64-PQFP programmed by Segger	U8	from Segger	
65	1	Crystal 18.432MHz, 30ppm 10pF, SMD 3.2x2.5 mm	Y3	535-10909-1-ND	Abrakon ABM8G-18.432MHZ-4Y-T3



LOWER CIRCUIT BOARD

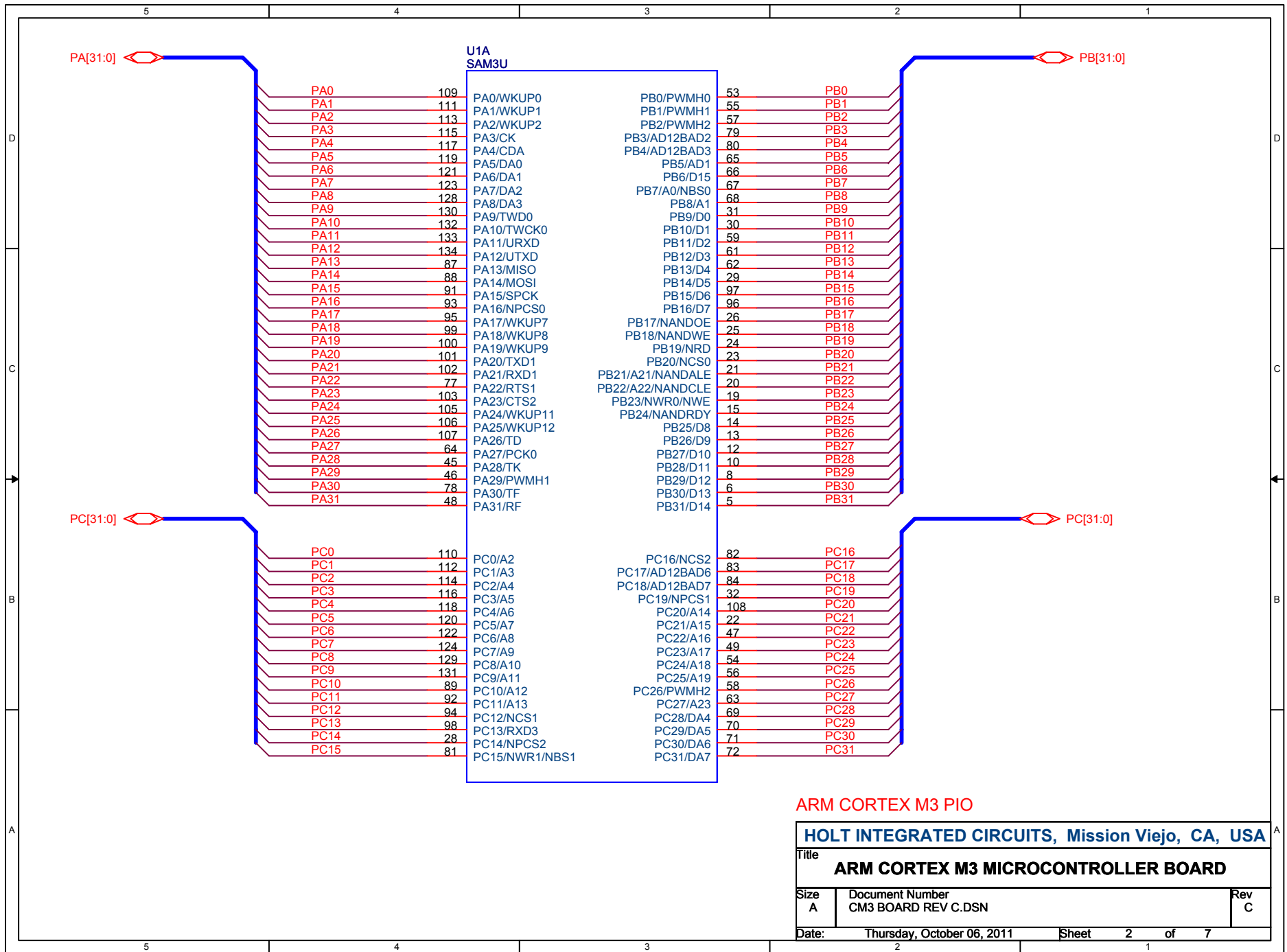


STACKING UPPER CIRCUIT BOARD

J3,J4 & J5 ARE DUAL-ROW STACKING RECEPTACLES (LOWER BOARD) AND HEADERS (UPPER BOARD).

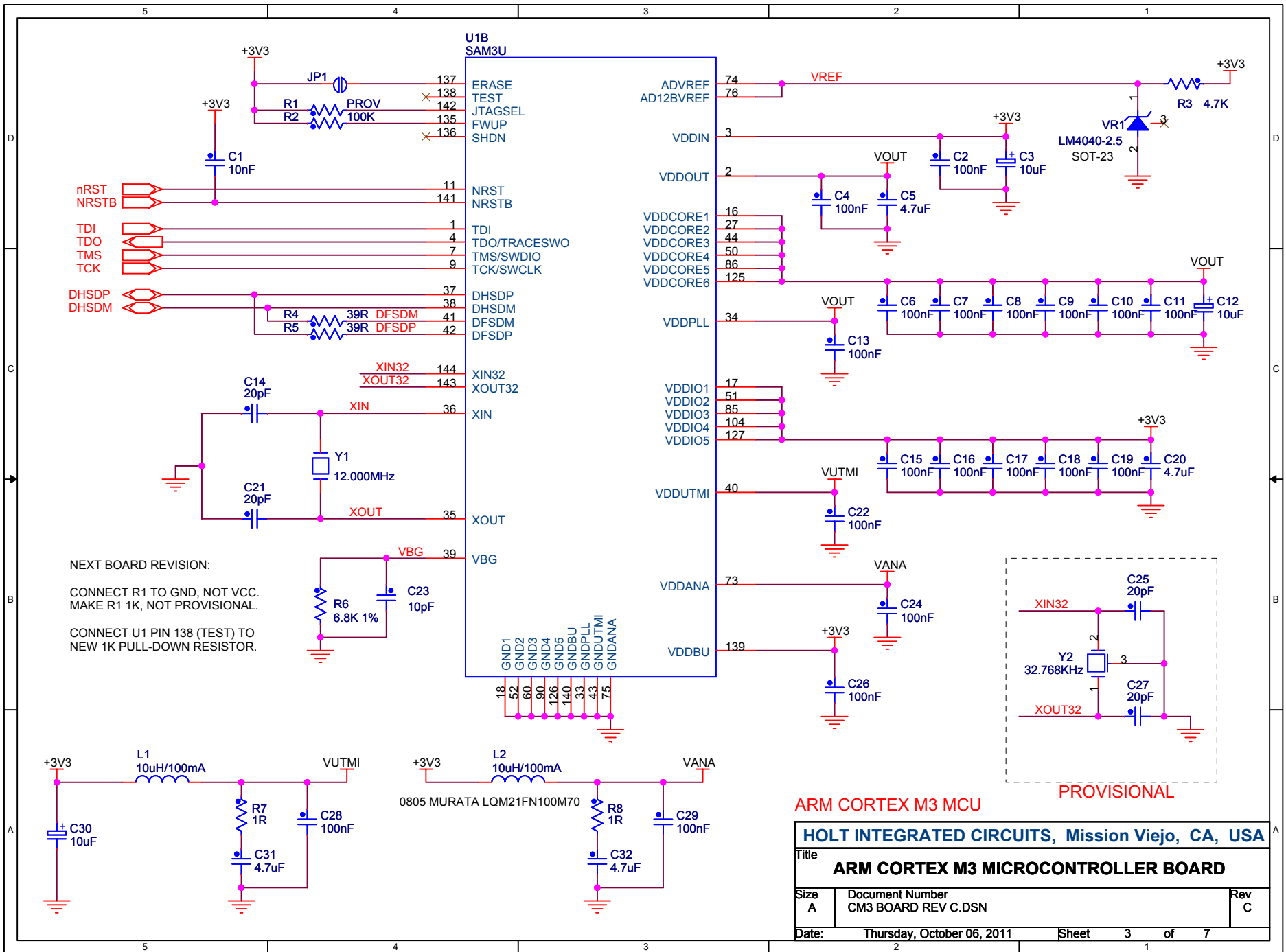
HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title		
ARM CORTEX M3 MICROCONTROLLER BOARD		
Size	Document Number	Rev
A	CM3 BOARD REV C.DSN	C
Date:	Thursday, October 06, 2011	Sheet 1 of 7

BUTTONS

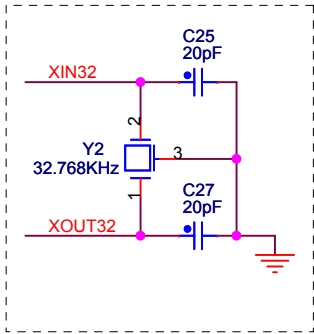


ARM CORTEX M3 PIO

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title ARM CORTEX M3 MICROCONTROLLER BOARD		
Size A	Document Number CM3 BOARD REV C.DSN	Rev C
Date:	Thursday, October 06, 2011	Sheet 2 of 7



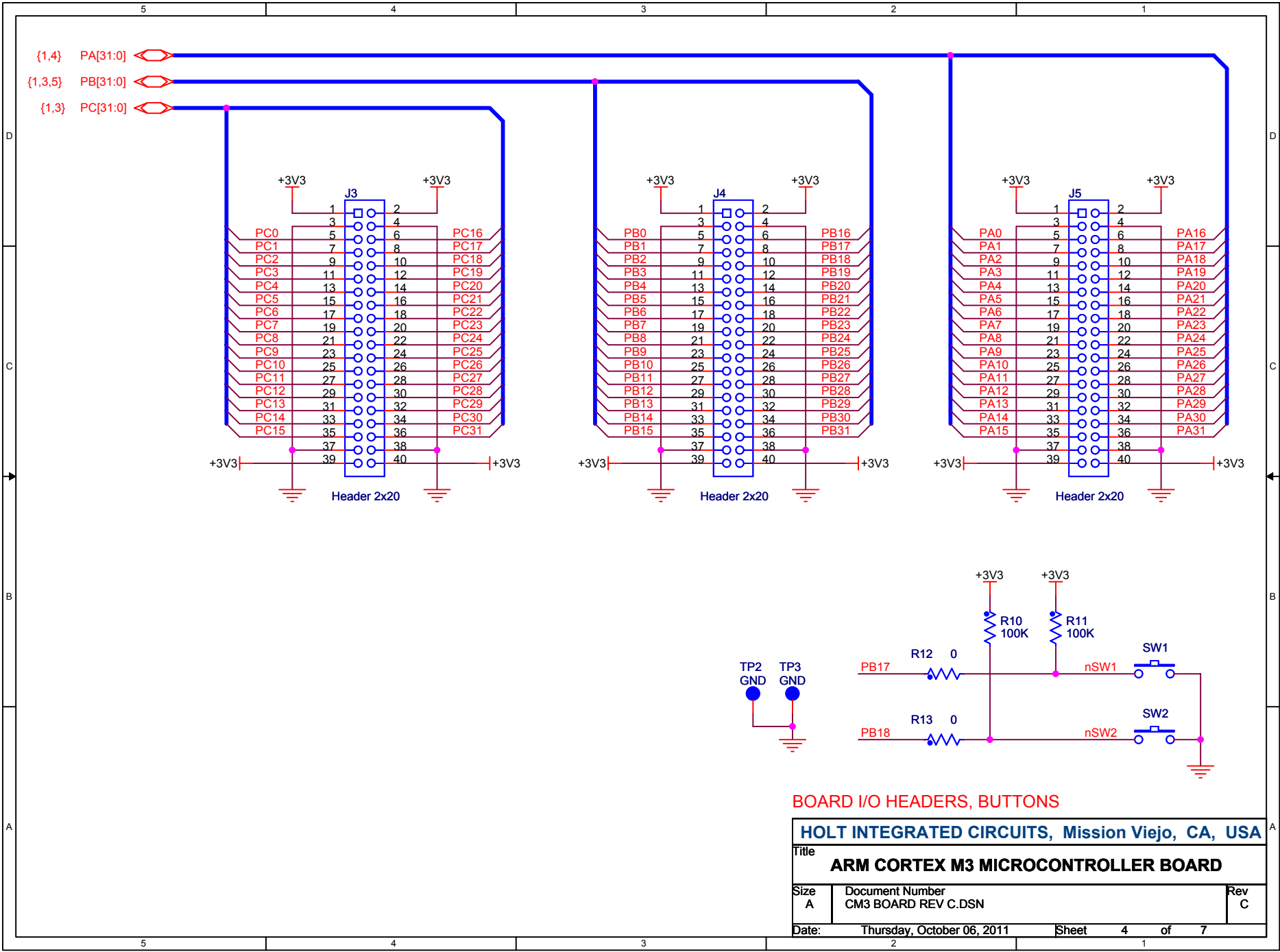
NEXT BOARD REVISION:
CONNECT R1 TO GND, NOT VCC.
MAKE R1 1K, NOT PROVISIONAL.
CONNECT U1 PIN 138 (TEST) TO
NEW 1K PULL-DOWN RESISTOR.



ARM CORTEX M3 MCU

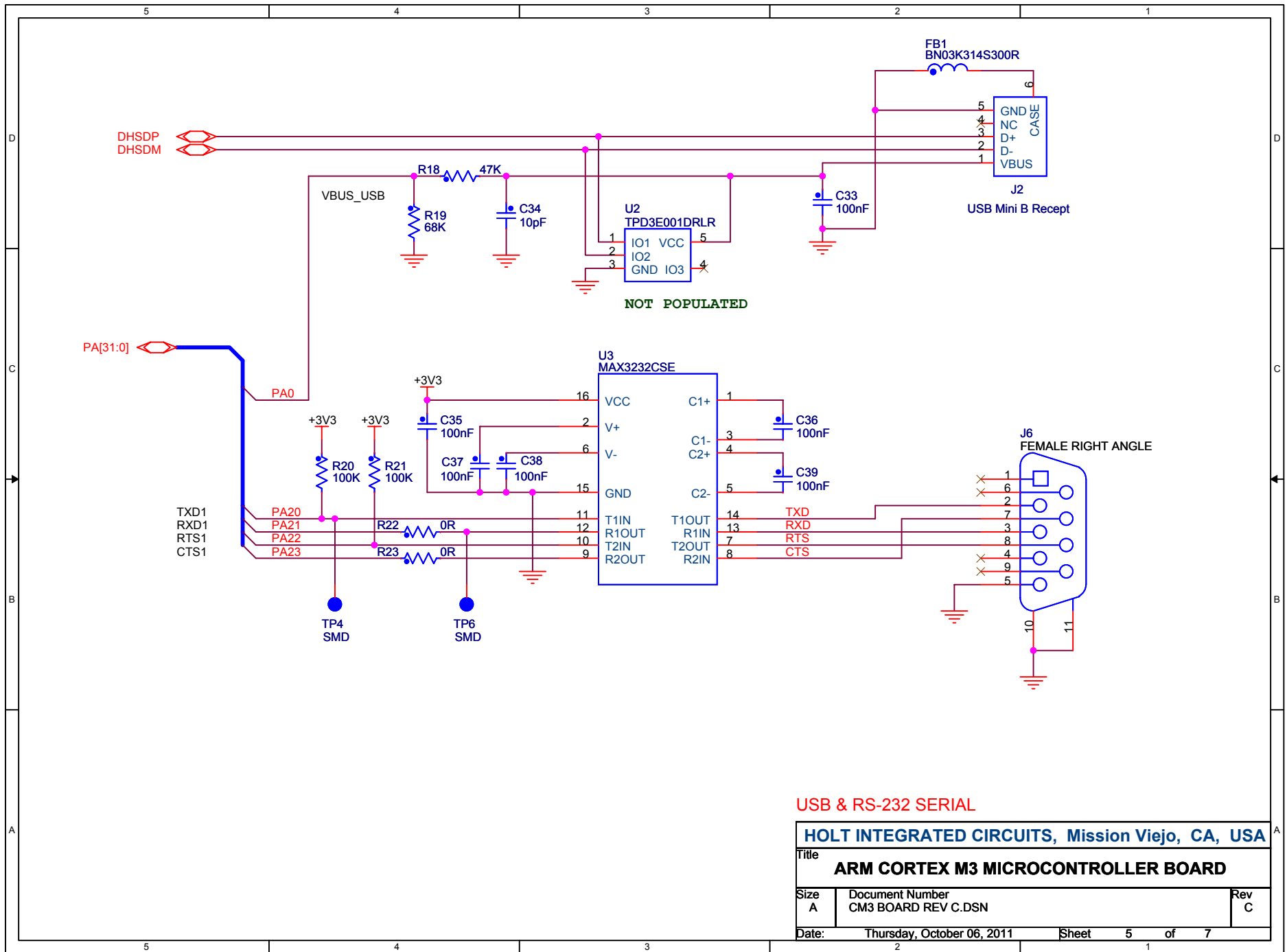
HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA

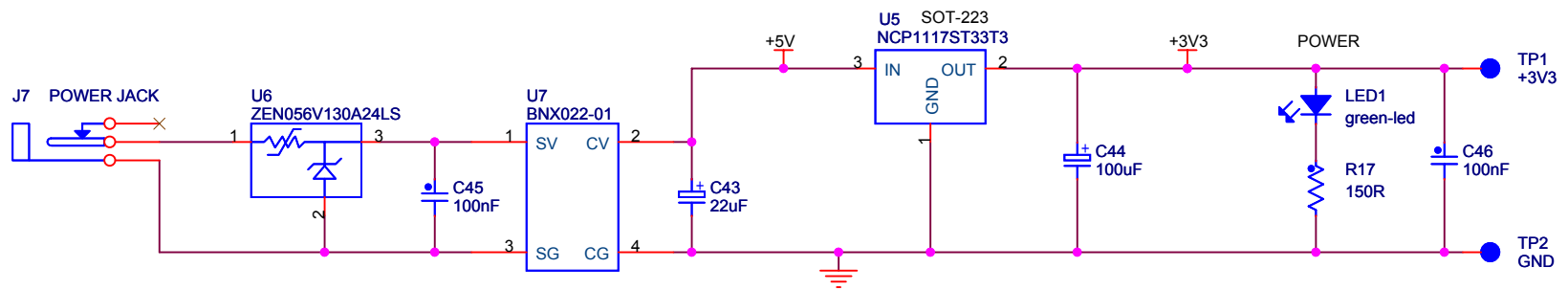
Title		
ARM CORTEX M3 MICROCONTROLLER BOARD		
Size	Document Number	Rev
A	CM3 BOARD REV C.DSN	C
Date:	Thursday, October 06, 2011	Sheet 3 of 7



BOARD I/O HEADERS, BUTTONS

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title ARM CORTEX M3 MICROCONTROLLER BOARD		
Size A	Document Number CM3 BOARD REV C.DSN	Rev C
Date: Thursday, October 06, 2011	Sheet 4	of 7





POWER SUPPLY

HOLT INTEGRATED CIRCUITS, Mission Viejo, CA, USA		
Title ARM CORTEX M3 MICROCONTROLLER BOARD		
Size A	Document Number CM3 BOARD REV C.DSN	Rev C
Date:	Thursday, October 06, 2011	Sheet 6 of 7

