ADK-175 User's Guide – HI-3110 Evaluation Board

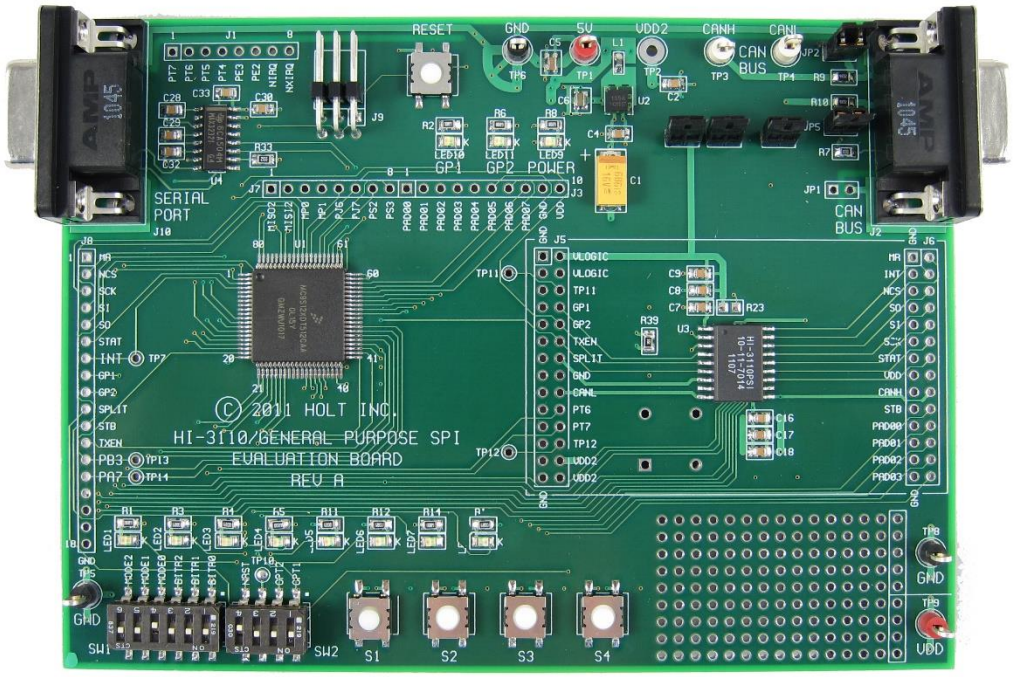
April 2020

REVISION HISTORY

Revision	Date	Description of Change
AN-175 Rev. New	08-30-11	Initial Release
Rev. A	01-15-13	USB to Serial, External SPI Interface and Two node demonstration.
Rev. B	04-20-2020	Improve board schematic diagram resolution. Update Freescale Development Tools information. Update document to new format.

Introduction

The Holt HI-3110 ARINC 825 CAN BUS Controller Evaluation Board can be used to evaluate some of the features of the HI-3110 CAN Controller IC with integrated transceiver. A Freescale MC9S12XDT512 microcontroller with 512K flash and 20K RAM communicates with the HI-3110 through the SPI interface. The board includes switches and LEDs to help navigate the operating modes and confirm data and status information. A Serial UART port is provided to allow debug and data messages to be sent to a PC using any terminal program such as Hyper-terminal.



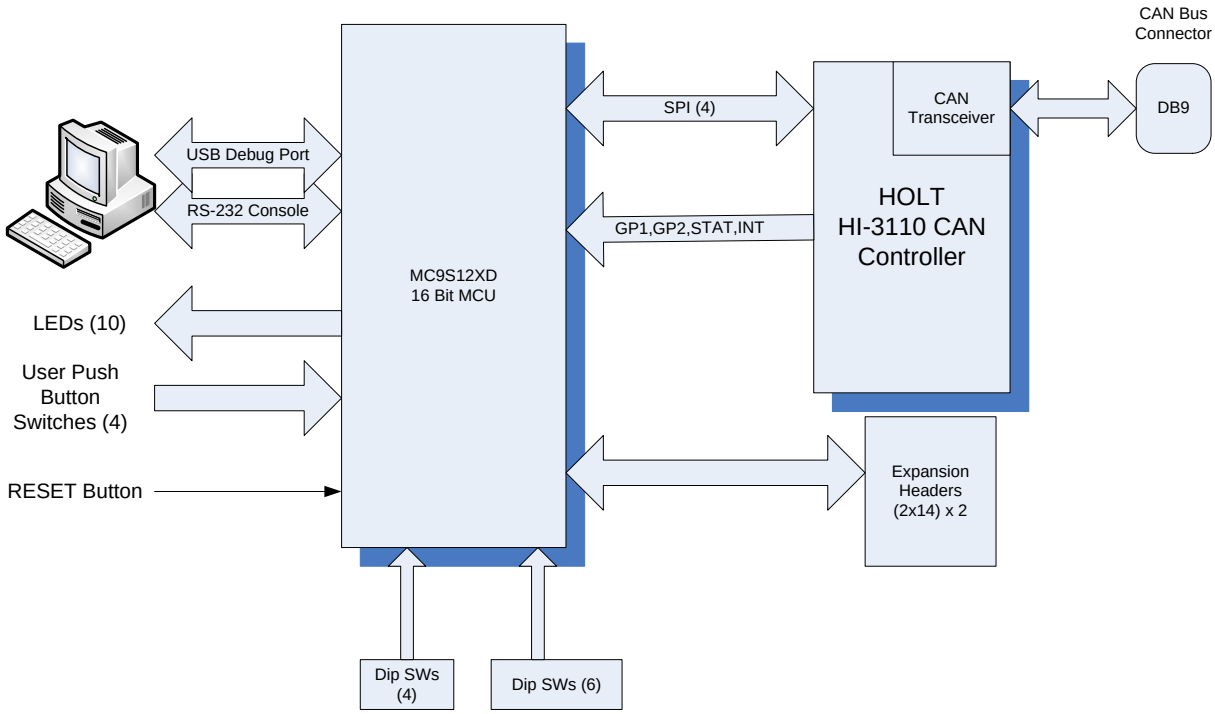
KIT CONTENTS

- This Users Guide.
- HI-3110 Evaluation Board
- HI-3110 Data Sheet
- RS-232 Serial Cable
- USB to Serial Adapter
- Codewarrior™ Demo Software Project

Demonstration Features

- Transmit Modes for Standard and Extended Frames
- Receive Modes, polled and interrupt driven
- CAN bus communication using two nodes
- Monitor Mode
- LoopBack Mode
- Sleep Mode with wake-up from Button press or CAN data
- Selectable Bit Rates from 83.3KBPS - 1MBPS
- Print out the CAN buffers (64) on the serial console
- Print out the status registers on the serial console
- Print out the Transmit History FIFOs on the serial console
- Print out the Error Status registers on the serial console
- Configuring the INT, STAT and GP1 and GP2 outputs
- SPI clock frequency selectable from 625 KHz – 20 MHz.
- Test Mode for cycling GPIOs and LEDs as a board check
- CAN Filters and Masks

Demonstration Set-up



The board requires a +5V (0.5A) power supply applied to TP1 and Gnd to TP6. There are two ways to connect the CANL and CANH signals to the CAN Bus. The CANL and CANH Bus signals are available on the two test points on the top right side of the board or from the DB9 connector.

Power On Reset

For normal operation ensure SW2 – 4 (MRST) is in the open position otherwise the MCU will be held in the reset state. The purpose of this MRST dip switch is to allow easy interfacing of an external MCU to the HI-3110 such that the SPI signals from the Freescale MCU will be forced into high impedance so as not to conflict with an external MCU. For normal operation keep this switch open and use the RESET button to reset the MCU during testing.

For Modes 1-6 the MCU will initialize the HI-3110 and check the STATF status register for the default 0x82 value, if the status value does not match 0x82 the program will turn on the Red LED8 and halt.

CAN BUS Bit Rate selection

The Bit Rate can be selected by configuring the 3 dip switches in SW1. The program fetches the switch positions once after a power up.

Bit Rate Selection

S3	S2	S1	BIT RATE
0	0	0	1 MBPS
0	0	1	500 KBPS
0	1	0	250 KBPS
0	1	1	125 KBPS
1	0	0	83.3 KBPS

Mode Selection

MODE 2	MODE 1	MODE 0	MODE
0	0	0	Board Test
0	0	1	Transmit Standard Frames
0	1	0	Receive in polling mode
0	1	1	Monitor
1	0	0	Loop Back
1	0	1	Transmit Extended Frames
1	1	0	Receive in interrupt mode
1	1	1	Serial Console Commands

SW2 OPT1 switch (Receiver data frame compare)

Receiver Modes:
If OPT1 is in the open position the receiver mode will check the data frame for an incrementing 32-bit value. If the data doesn't match the expected value from the previous transmission it will turn on the Red LED8 and halt.

Transmitter Modes:
If OPT1 is in the open position a 400us delay will be inserted between each transmission.

SW2 OPT2 switch

OPT2 is also used in the receiver mode. It enables the FILTON bit(bit4) in the CTRL1 register for CAN ID filtering. See the Receive Mode description in the software description section for more details.

Buttons1 – Button4 (SW1-SW4)

Depending on the mode these buttons will perform specific tasks.

Serial Port Configuration

The serial port is optional and provides access to additional status and data information:

Software Revision
Current Mode
CAN Frame data
CAN History FIFO data
Status Registers data
Sleep Mode status
Serial Commands Menu

A simple menu of commands is provided for resetting the HI-3110, reading and displaying the HI-3110 status registers, and a demo how to pass input strings to the program. There are provisions for additional user code to be implemented so custom commands can be developed and easily integrated.

UART Serial Port Setup: 115200 Baud, 8 bits, No Parity, No handshaking.

A USB to Serial adapter is provided in the kit which can be used with PC's that don't have a serial port.

At power on reset (POR) the revision of the program will be displayed on the LEDs for two seconds in binary format where LED1 is the LSB. After two seconds they will be turned off then LED7 will flash every second as the main MCU "live" operating indicator. A message will also be sent to the Serial Port . For Receive mode it will be shown as:

```
Holt HI 3110 Demonstration Software Revision: xx
Receive Mode polled
```

Board Jumpers

JP1 - CAN BUS connector shield
JP2 - CANH 60 ohm termination.
JP3 - VDD jumper to U3 VLOGIC. Removing JP3 can be used to measure ICC.
JP4 - SPLIT termination voltage filter.
JP5 - CANL 60 ohm termination.
JP7 - VDD jumper to U3 VDD. Removing JP7 can be used to measure ICC or to operate the logic section of the HI-3110 at 3.3V while keeping 5V on the VDD pin for the integrated CAN transceiver. See the HI-3110 Split Voltage Operation section below.

Using the HI-3110 with an external MCU or FPGA

To use the HI-3110 with an external MCU or FPGA close the RESET DIP switch 4 in SW2. This will hold the Freescale MCU in the reset state and keep the SPI signals from the Freescale MCU in tri-state. This allows the SPI signals to be driven by an external source without having to remove the Freescale MCU on the board. The SPI signals to the HI-3110 are available on both J8 header rows or J5 and J6. Refer to the schematic for the pin assignments. Connect MR, nCS, SO, SI, SCK and optionally STAT and INT to the external MCU or FPGA. The HI-3110 supports either 5V or 3.3V logic levels depending if 5V or 3.3V is applied to the Vlogic pin. 5V is needed on the VDD pin for the transceiver outputs to meet ISO-11898-5 voltage levels.

HI-3110 Board Dual Voltage Configuration

It is possible to run the digital logic of the HI-3110 on 3.3V while keeping 5V on the integrated CAN transceiver on the VDD pin-11. The Freescale MC9S12XD family can operate at either 3.3V or 5V so this microcontroller can be used to demonstrate this. To setup the evaluation board for dual voltage operation follow the simple setup instructions below:

Setup instructions:

1) Remove JP7 and connect the right side of the JP7 jumper post to TP2 VDD2. **Do not** re-install the jumper shunt.

2) Connect a 3.3V power supply to the board TP1 VDD.

3) Connect a 5V power supply to the board TP2 VDD2.

Users Guide

Mode-0: Board Test

In this mode the MCU performs a walking pattern on the GPIO pins and cycles turning on all the LEDs. See the boardtest.c module for details and other options available.

Mode-1: Transmit Mode Standard Frames

This transmits a standard CAN frame with an incrementing 32-bit value in the first 4 bytes of the data – field, the last four bytes will be 00. After RESET press the Button-1 to start the transmissions. A portion of the data field will be copied to the LED1-LED4's. They can be seen counting in this mode.

Press Button-1 to stop transmitting and print out the status registers on the serial console.
Press Button-2 to pause the transmission. Press Button-2 again to transmit one frame again.
Press Button-3 to print out the Transmit History FIFO contents on the serial console.
Press Button-4 to resume transmissions.
OPT1 switch= open, Inserts a 400us delay before each transmission.

Mode-2 Receive Mode using Polling method

To use Receive mode with an external node transmitter, set the OPT1 switch to the closed position to disable the Receiver data frame compare feature. This will allow any data frame to be accepted.

LED1-4 will reflect the lower nibble of byte3 of the data-field.

CAN Error Detection

While receiving frames if there are any errors detected in the HI-3110 registers, TEC, REC or ERR, they will be automatically logged on the Serial Port.

For example if ERR has a 01 value it would be shown as: ERR: 01. The other two errors will be shown similarly.

Button-1 – Stop receiving CAN messages and print the last 64 CAN frames and the status registers on the serial console. Press Button-4 to resume receiving CAN messages. See a partial listing of this output below which shows the last two frames.

```
CAN Frame 62: 00 77 60 47 20 00 00 08 00 00 22 28 04 05 06 07
        Standard  Filter:0  TTU:77  TTL:60  BaseID:239
        [r0,r1,x,x][DLC]: 08  Data-Field: 00 00 22 28 04 05 06 07
```

```
CAN Frame 63: 00 78 61 47 20 00 00 08 00 00 22 29 04 05 06 07
        Standard  Filter:0  TTU:78  TTL:61  BaseID:239
        [r0,r1,x,x][DLC]: 08  Data-Field: 00 00 22 29 04 05 06 07
```

Button-2 – Stop receiving CAN messages and print the HI-3110 status registers. Press Button-4 to resume CAN messages.

CTR0 = 00
CTR1 = 00

BTR0 = 40
BTR1 = A7
TEC = 00
REC = 00
MESSTAT=00
STATF = 00
ERR = D8
INTF = 00
INTE = 82
STATFE= 00
GPINE = 00
TIMERUB=98
TIMERLB=95

Button-3 – Put the HI-3110 into Sleep Mode. Press Button-4 or send CAN messages in Sleep mode to terminate and re-enter Normal mode. A message on the serial console will indicate the Sleep Mode and the source to the wake-up event.

```
Holt HI-3110 Demonstration Software Revision: 09
Receive Mode polled

Entering SLEEP Mode

Press Button-4 to wake-up.

Normal Mode Resumed from Sleep Mode

Entering SLEEP Mode

Press Button-4 to wake-up.

Normal Mode Resumed from Monitor Mode
REC: 01 ERR: 04
```

The last line shows a BUS error that was caused when the HI-3110 was in Sleep Mode and woke up due to a CAN message coming in.

CAN bus communication using two nodes

Two-node communications can be demonstrated using two HI-3110 evaluation boards. One board is configured in Mode-1 as the transmitter and the second board is configured in Mode-2 as the Receiver.

Two-node demo setup:

1. Connect CANH (TP3) from board-1 to CANH of board-2 together using a clip lead. Connect CANL (TP4) signals together on each board using a clip lead. Optionally, a 9-pin CAN cable can also be used in place of the clip leads (not provided in the kit).
2. Ensure the bit rate DIP switches S1, S2 and S3 are set for the same bit rate on both boards.
3. Set board-1 to Mode-1 as the Transmitting node. Mode-1 transmits an incrementing data pattern starting from count zero.
4. Set board-2 to Mode-2 as the Receiver node. Set OPT1 switch to the open position to enable the Receiver to perform 32-bit data frame compares.
5. Power up both boards and optionally press the Reset button on both boards.
6. To start the data transmissions press S1 on board-1.

Board-1 begins to transmit data frames and the Receiver board-2 will receive the data frames. The Receive board LEDs 1-4 will reflect the lower nibble of byte3 of the data-field.

When OPT1 switch is open, The Receiver program will compare the 32-bit counter value in the data-field and turn on the RED LED8 and halt if it does not match the expected value. To exit from the halt condition, press the RESET button. To repeat the test after a data mismatch, reset both boards and repeat the test by pressing the S1 on board-1 again. The data-field compare function can be turned off by closing the OPT1 switch. With the compare function turned off, the Receiver will continue to receive data frames and update the LEDs 1-4 from the lower nibble of byte3 of the data-field.

Mode-3 Monitor Mode

Monitor mode is a simple version similar to Receive mode but just captures the CAN data without providing an acknowledge bit on the Bus.

Press Button-1 to stop receiving CAN messages and print the last 64 CAN frames and the HI-3110 status registers on the serial console.

Press Button-4 to resume reception of CAN Messages.

Mode-4 LoopBack Mode

LoopBack mode combines a Transmitter and Receiver function similar to the normal Transmit and Receive modes. The Transmitter will transmit the 32-bit incrementing counter in the data-field and the receiver looks and compares against this data pattern and will turn on the RED LED the same way as in normal Receive Mode. No actual data is transmitted to the Bus in LoopBack mode.

Button-2 - Stops transmitting CAN data and prints out the contents of the Transmit History FIFO as shown below. Press Button-4 to continue.

```
History FIFO: 04 86 00
History FIFO: 04 87 00
History FIFO: 04 87 81
History FIFO: 04 88 03
History FIFO: 04 88 84
History FIFO: 04 89 06
History FIFO: 04 4D E2
```

Button-1 – Stop receiving CAN messages and prints the last 64 CAN frames and the HI-3110 status registers on the serial console. Press Button-4 to resume receiving CAN Messages.

Mode-5 Transmit Extended Frames

This is the same as Transmit Mode-1 but with Extended Frame format.

Mode-6 Receiver Interrupt Mode

This is very similar to normal Receive mode-2 but without Sleep Mode. The main purpose is to demonstrate how to use interrupts to received the CAN Bus data in the background.

Press Button-1 to stop receiving CAN messages and print the last 64 CAN frames and the HI-3110 status registers on the serial console.

Press Button-4 to resume reception of CAN messages.

Mode-7 Serial Commands Menu

After a power on reset, a simple text menu will be displayed on the serial console:

```
************** HOLT HI-3110 Serial Console Commands

 Input String Demo          0
 Reset                      1
 Status Register Read       2
 Filter Incrementing        3

 Enter Selection:2

Enter "1" to generate a hardware reset to the HI-3110.

Enter "2" for the status register read and display:

CTR0 =80
CTR1 =00
BTR0 =00
BTR1 =00
TEC =00
REC =00
MESSTAT =00
STATF =00
ERR =00
INTF =00
INTE =82
STATFE =00
GPINE =00
TIMERUB =0D
TIMERLB =66
```

Software Description

The software project is built with Freescale's Codewarrior version 5.9.0 using the free version which is limited to 32K code size. The current code size is approximately 10.5K. The main functions are in main.c and the lower level HI-3110 drivers are in the 3110Driver.c file. Some program flow charts are provided to help clarify some of the program.

Project Files

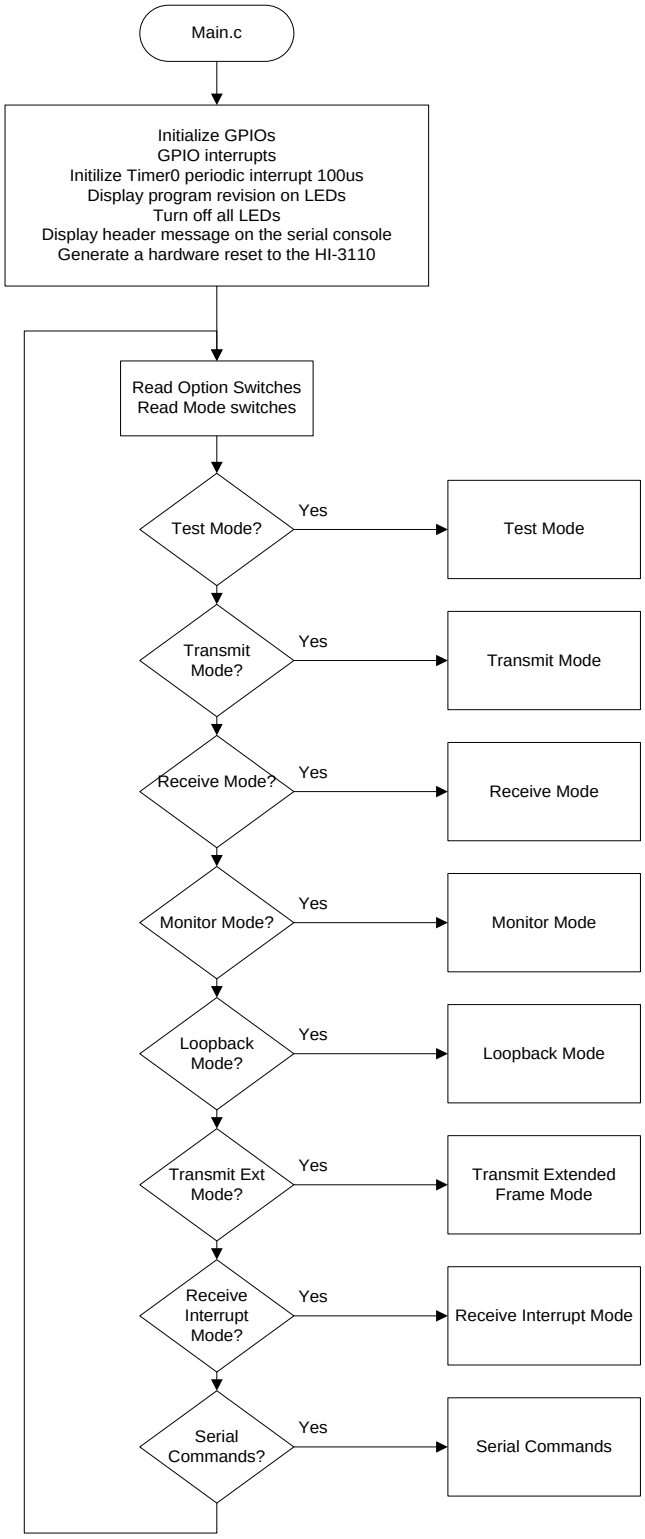
Source Files

main.c	Main code
3110Driver.C	SPI low-level drivers for the HI-3110
Peripherals.c	GPIO, PLL frequency setup and SPI configuration
BoardTest.c	Board Test functions
Uart.c	Low-level UART drivers
datapage.c	Freescale IDE support file

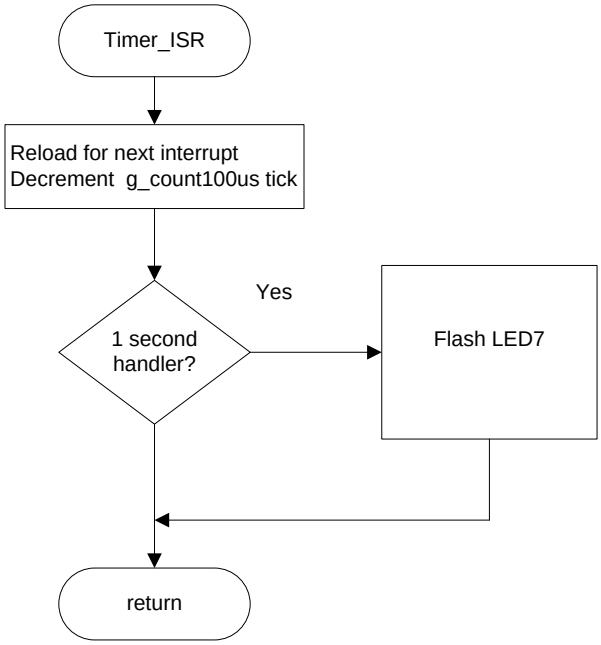
Include Files

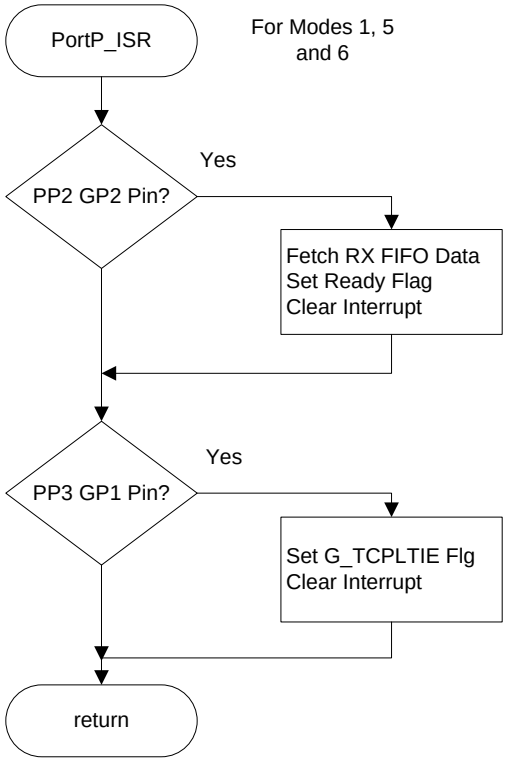
Main.h	
3110Driver.h	HI-3110 header file
Peripherals.h	Peripherals header file
BoardTest.H	Board test header file
Uart.h	Uart header file
Common.h	Common defines for the project
Derivative.h	Freescale IDE support file
Mc9s12xdt512.h	Freescale IDE target part support file

Main Flow Chart

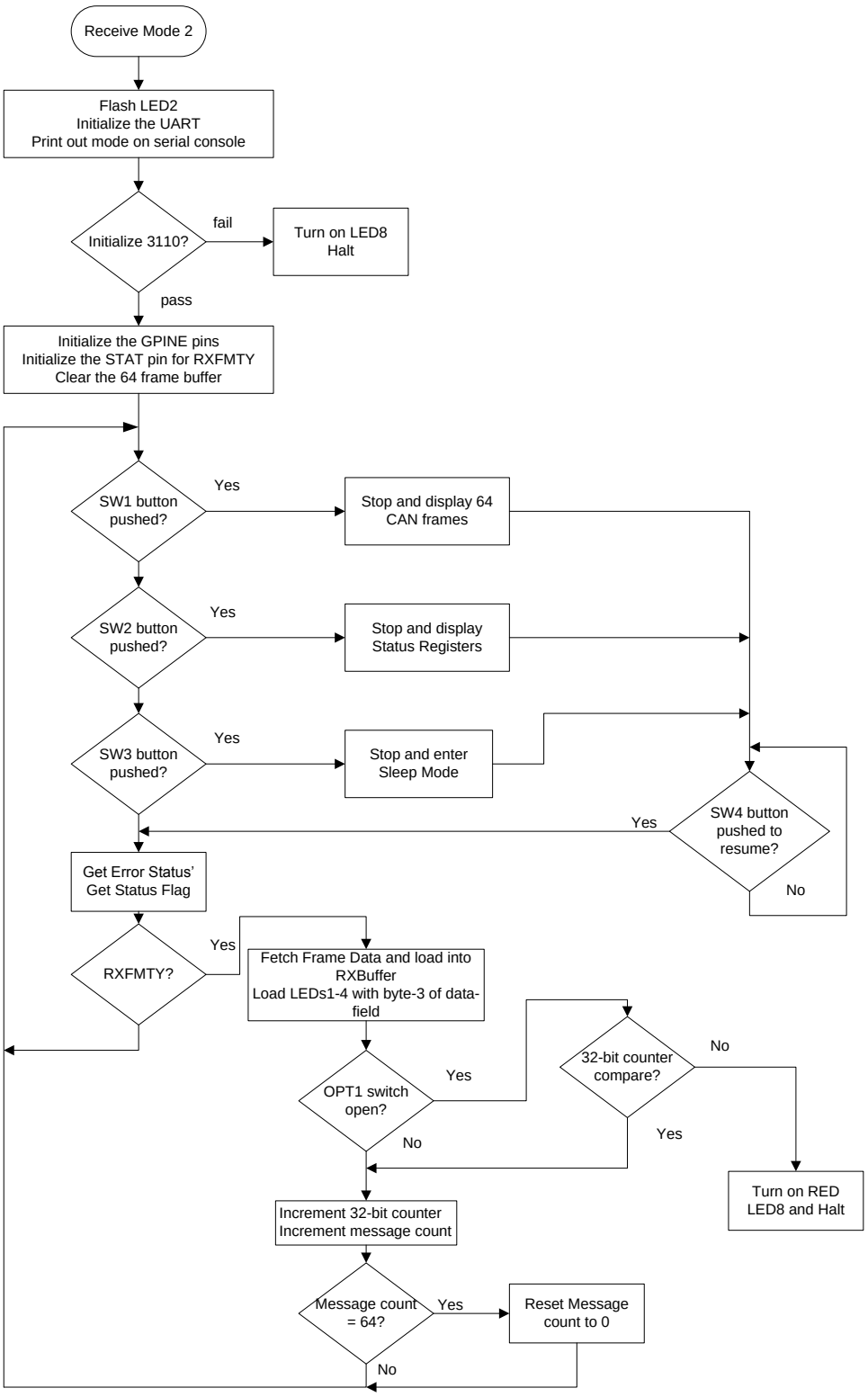


Interrupt Flow Charts

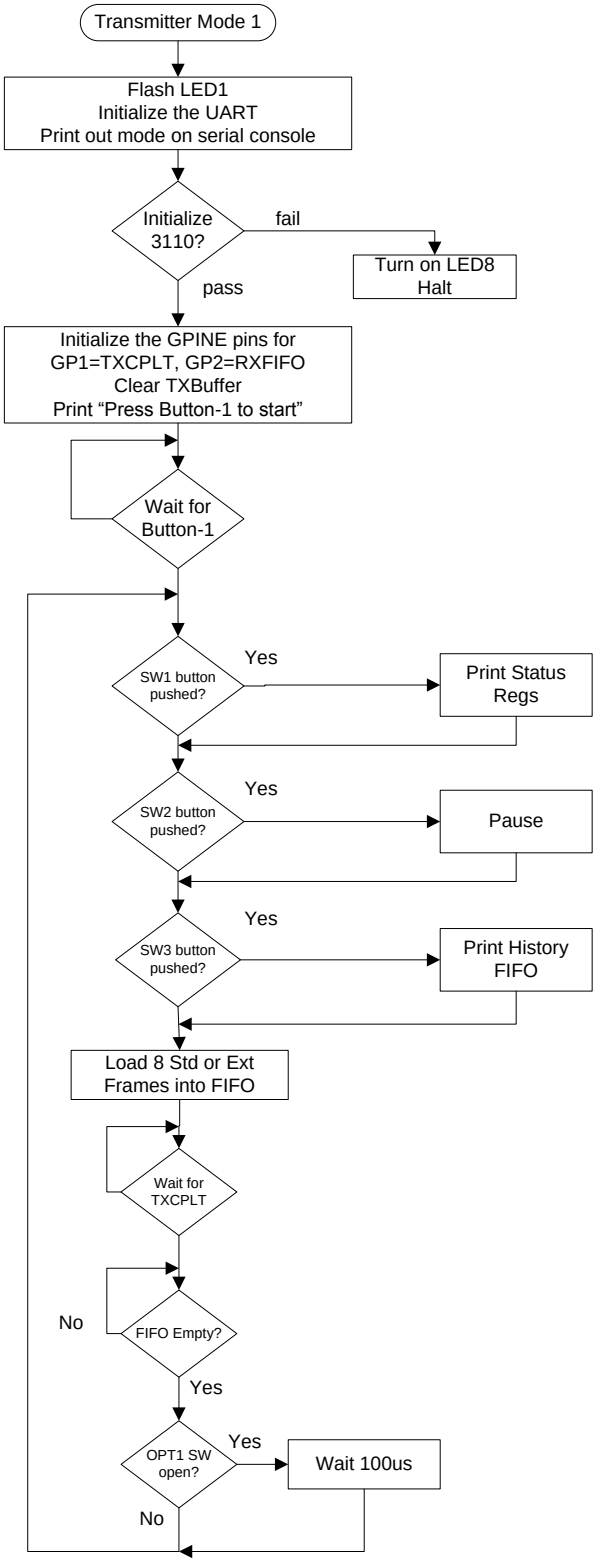




Receiver Mode 1 Flow Chart



Transmitter Modes 1 and 5 Flow Chart



MCU Clock and SPI Frequency

The Freescale MC9S12XDT512 (MCU) uses a 4MHz crystal for operation and the built-in PLL is used to multiply 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:

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

The SPI frequency is set by this line of code:

```
SPI0BR = SPI_5MHZ;     // SPI CLK = 5MHz (see
                       // "peripherals.c"  for other rates)
```

Timer Interrupt and timings

A 100us Timer0 interrupt supports basic timings. A number of predefined constants are provided to allow some common delays passed into the Delay100us() function:

```
#define K_1MS  10
 #define K_10MS 100
 #define K_100MS  1000
 #define K_1SEC  10000


Usage: Delay100us(K_1SEC); // delay for one second
```

A one second interrupt handler in the TIMER_ISR is provided. Any code placed here will automatically get executed every second.

```
if (!count100us)
    {       // 1 second scheduler
      count100us = K_1SEC;
      if(ON==g_ledFlashBool)            // Flash the
         LED8 ^= TOGGLE;                // Alive 1 second blink
    }
```

HI-3110 Interrupts

The HI-3110 can be configured to output status on the INT, STAT, GP1 and GP2 pins which connect to the MCU PP0 – PP3 inputs. Interrupt handlers are provided for the GP1 and GP1 for servicing TXCPLT and RXFIFO respectively on the PP3 and PP2 pins which are enabled by calling PortP_Int().

```
Interrupt 56 void PORTP_ISR(void)
```

The PP2 interrupt handler is provided to interrupt when the 3110 receives a CAN Bus frame of data. This interrupt handler is utilized only in Receive Mode Interrupt Mode-6. Once an interrupt occurs the interrupt handler fetches the frame of data and stores the data into a global buffer (g_RXBuffer) and then sets a flag "g_RecRdy=TRUE". The main receiver function looks for this flag and fetches the new CAN data when this flag is set.

There are two other possible sources for hardware interrupts using the XIRQ and IRQ inputs reserved for possible future use.

3110Driver.c Drivers

All the HI-3110 SPI drivers are contained in this module for reading and writing to the control registers, status registers, filters/masks and FIFO's .
There are several functions to read in a single byte from the 3110 SPI port, write a command to the 3110 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 out a command plus one byte of data to the 3110 SPI port.

```
// Write SPI Command with a Value to HI-3110
void W_CommandValue (uint8 cmd, uint8 value){
  uint8 dummy;

  SPI0CR1 = SPI0CR1 & ~SPI0CR1_SSOE_MASK;   // disable auto /SS output, reset /SS Output
  SPI0CR2 = SPI0CR2 & ~SPI0CR2_MODFEN_MASK; // disable auto /SS output, reset SPI0 Mode
  SPI0_nSS = 0;                           // assert the SPI0 /SS strobe
  dummy = SPI0SR;                         // clear SPI status register
  SPI0DR = cmd;                           // SPI command
  while (!SPI0SR_SPIF);
  dummy = SPI0DR;           // read Rx data in Data Reg to reset SPIF
  dummy = SPI0SR;                           // clear SPI status register
  SPI0DR = value;                           // Reset values
  while (!SPI0SR_SPIF);
  dummy = SPI0DR;           // read Rx data in Data Reg to reset SPIF

  SPI0_nSS = 1;                             // negate the SPI0 /SS strobe
  SPI0CR1 = SPI0CR1 | SPI0CR1_SSOE_MASK;    // enable auto /SS output, set /SS Output
  SPI0CR2 = SPI0CR2 | SPI0CR2_MODFEN_MASK;  // enable auto /SS output, set SPI0 Mode Fault
}
```

This function is used to transmit a command byte followed by a multiple number of data bytes contained into the array passed by the pointer. In this case it's used to load the SPI Write Transmit FIFO command byte 0x12 and the frame data bytes.

```
void TransmitStandardFrame(uint8 *TXBuffer)
 {
 uint8 static ByteCount,dummy;

   SPI0CR1 = SPI0CR1 & ~SPI0CR1_SSOE_MASK;    // disable auto /SS output, reset /SS Output
   SPI0CR2 = SPI0CR2 & ~SPI0CR2_MODFEN_MASK;  // disable auto /SS output, reset SPI0 Mode
   SPI0_nSS = 0;                              // assert the SPI0 /SS strobe
   for(ByteCount=0; ByteCount< 13; ByteCount++)        // Transmit command=0x12 +  12
      {
         dummy = txrx8bits(TXBuffer[ByteCount], 1);  // Transmit the whole message,
                                                     // ignore return values
      }
   SPI0_nSS = 1;                              // negate the SPI0 /SS strobe
   SPI0CR1 |= SPI0CR1_SPE_MASK|SPI0CR1_SSOE_MASK;
   SPI0CR2 |= SPI0CR2_MODFEN_MASK;
 }
```

Al l the other functions in this module are based on and utilize these basic functions.

Init3110()

This function is provided to initialize the HI-3110 for the desired mode and bit rate.

```
uint8 Init3110(const mode, uint8 ttdiv, uint8 wakeup, uint8 reset, uint8 bor)
```

See the 3110Driver.h header file for the options available in the define statements.

Uart.c Serial Port (RS-232)

The drivers to support the serial port 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 Hyper-terminal or any other terminal program. It currently uses polling to determine when the data receive or transmit registers can be read or written.

GPI and GP2

The HI-3110 general purpose output pins can be configured to output status information described in the GPINE register. The program configures these differently depending on the mode and the LED10 and LED11 will turn on when these are at a logic low. GP1 and GP2 are configured to output TXCPLT and RXFIFO to generate interrupts.

LEDs LED1-LED8

The group of eight LEDs (LED1-LED8) are supported by a function in the program since LED1-LED4, LED8 are low true logic whereas LED5-LED7 are high true logic.

```
Usage: LED_CTL(LED_1,OFF);       // turns off LED1
Usage: LED_CTL(LED_1,ON);        // turns on LED1
```

TRANSMIT MODE

Select transmit mode from the SW1 dip switches and Press Button-1 after a POR to start transmitting frames. The transmit function configures the HI-3110 for normal mode at the CAN Bus rate according to the Bit Rate dip switches. A frame is composed of a predefined header "T8Header[]" and eight bytes for the data-field. For Mode- 5, Extended Frame Format a different header is used "T8HeaderExt". The program concatenates a 32-bit incrementing counter "BigCounter.Word32" to the header data and loads this into a transmit buffer "TXBuffer" before loading the FIFO with the frame before each transmission. For debugging purposes there is a special function "getRegStatus()" which will read out the status registers and load them into an array "DebugArray[]" for viewing if Button-1 is pressed. The transmit function writes eight frames of data to the FIFO's then waits for one transmission completion then waits again until the FIFO becomes empty before attempting to reload the next eight frames.

RECEIVE MODE

Select receive mode from the SW1 dip switches and the program will go directly into this mode after a power on reset. The receive mode reads in a frame by first examining the receive FIFO empty bit in the status register. Received frames are compared with the expected 32-bit counter valued after each transmission if the OPT1 switch is in the open position. If the 32-bit counter value does not compare the RED LED8 will be turned on and the program will halt. Close the OPT1 switch to disable pattern checking.

Received frames are stored into a 64 deep array "RXBuffer[64][16]" which provides a convenient way to view the data in the debugger or by outputting the data on the serial console. When Button-1 is pressed the program will halt receiving CAN data and print out the last 64 frames of data captured in the array on the Serial Port. See the example of the print out in the users guide section.

Filtering on ID

If two boards are used one can be setup to transmit and the other to receive. If the receive board has the OPT2 switch open it will filter the frame data based on the Filter and Mask values stored in the Filter and Masks arrays in the 3110Driver.c module. When the OPT2 switch is open the FILTON bit-4 in the CTRL1 register for CAN ID filtering is set in the Init3110() function.

```
 // CAN Bus acceptance filters
unsigned const char Filters[8][7] ={                    // Filter data
   {W_FILTER0, 0x47,0x00, 0x00, 0x00, 0x00,0x00},      // Filter 0
   {W_FILTER1, 0x47,0x00, 0x00, 0x00, 0x00,0x00},
   {W_FILTER2, 0x47,0x00, 0x00, 0x00, 0x00,0x00},
   {W_FILTER3, 0x47,0x00, 0x00, 0x00, 0x00,0x00},
   {W_FILTER4, 0x47,0x00, 0x00, 0x00, 0x00,0x00},
   {W_FILTER5, 0x47,0x00, 0x00, 0x00, 0x00,0x00},
   {W_FILTER6, 0x47,0x00, 0x00, 0x00, 0x00,0x00},
   {W_FILTER7, 0x47,0x00, 0x00, 0x00, 0x00,0x00},      // Filter 7
   };

// CAN Bus acceptance masks
unsigned const char Masks[8][7] ={                      // Mask data
   {W_MASK0, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},        // Mask 0
   {W_MASK1, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},
   {W_MASK2, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},
   {W_MASK3, 0x00,0x00, 0x00, 0x00, 0x00,0x00},  // Accept everything with
filter 3
   {W_MASK4, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},
   {W_MASK5, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},
   {W_MASK6, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},
   {W_MASK7, 0xFF,0x00, 0x00, 0x00, 0x00,0x00},         // Mask 7
   };
```

This example code is setup to filter on ID28-ID20 = 0x47. The transmitter already transmits this value so by default the messages will be received. All other messages other than 0x47 will be except by filter-3. Refer to the datasheet page 38 and example code for the bit assignments of the other 5 bytes in these arrays.

Freescale MC9S12XDT512xxx Development Tools

Use CodeWarrior version 5.1 available on the NXP website from the Software Center location.

https://community.nxp.com/videos/5109

Select IDE – Debug, Compile and Build Tools. From this page click on Download (it will show a newer version 5.2). Select the Previous tab to download version 5.1. Version 5.2 requires adding missing derivative files and is not officially supported by NXP with the workaround. See NXP community forums for more information on this topic.

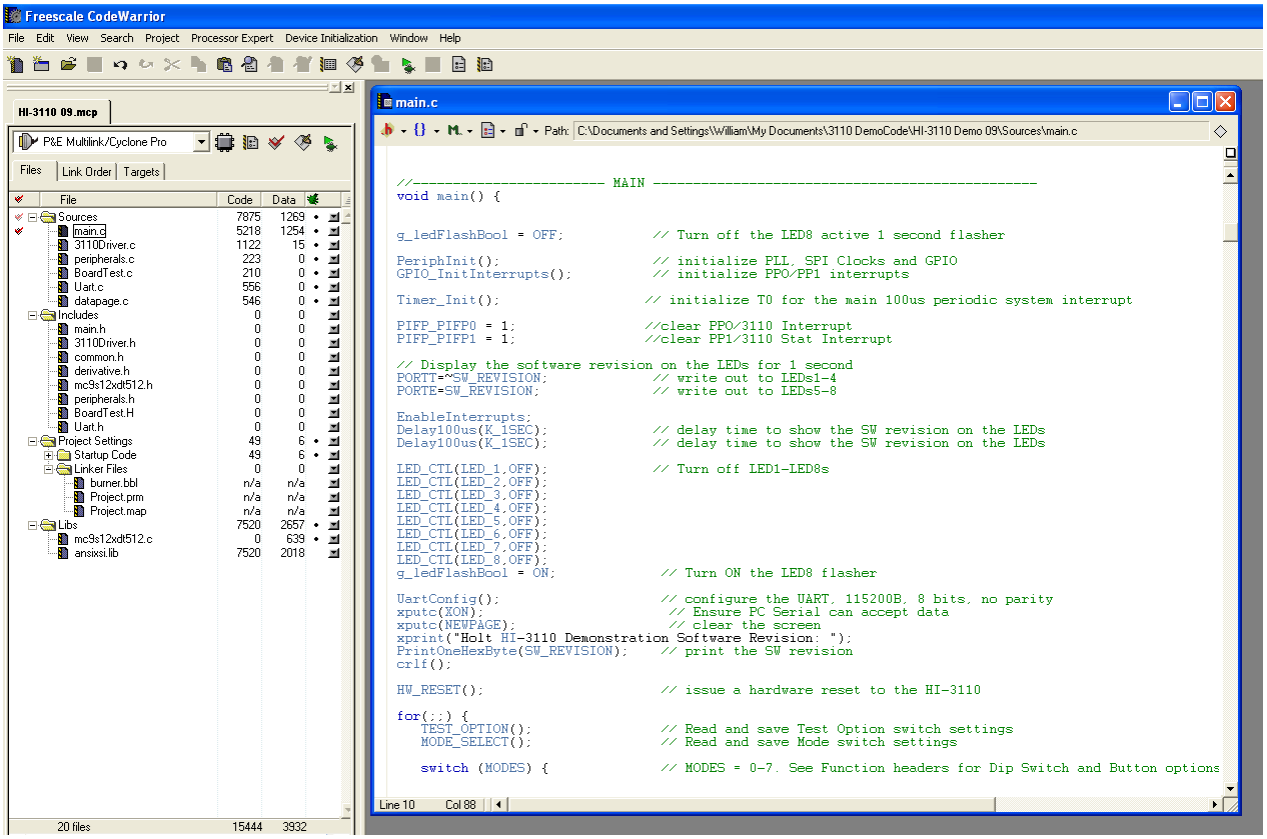
The USB debugger tool used on Holt demo boards supporting the Freescale MC9S12XD processor family for PE Micro USB debugger tool is now obsolete. Some Holt documents may still refer to the old debugger shown below: PN USBMULTILINKBDME

A suitable replacement is the PE Micro USB Multilink Universal.

A more advanced option is the PE Micro USB Multilink Universal FX

These can be ordered directly from PEMICRO.com or from Digi-Key.

Holt HI-3110 project loaded with Codewarrior 5.9.0



SUMMARY

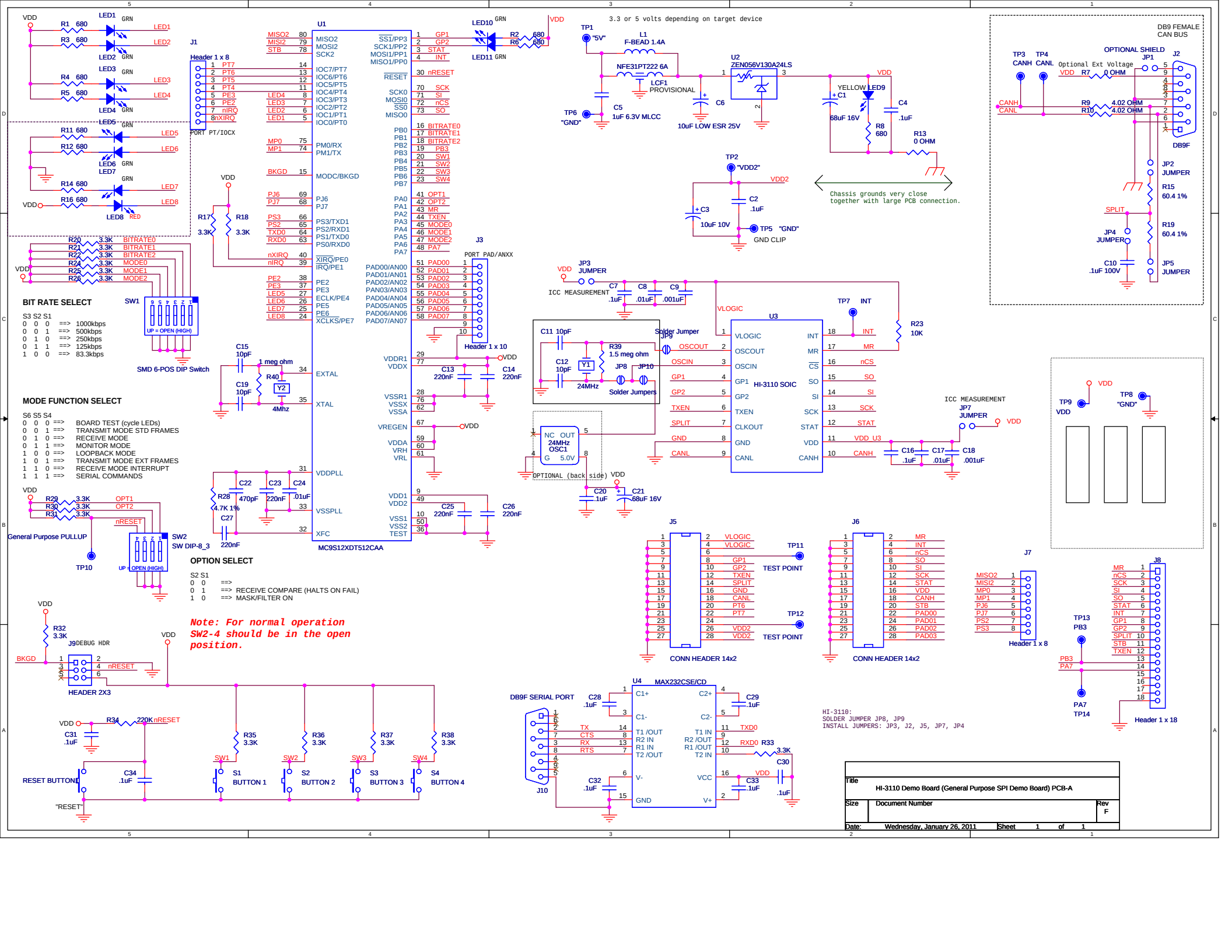
This Users Guide explains the features and capabilities of the HI-3110 demo software. To learn more about the HI-3110 refer to the HI-3110 data sheet. This document and the demo software project are included on the CD-ROM.

References:

http://www.holtic.com/

http://www.holtic.com/category/409-arinc-825-avionics-can-controller.aspx

Bill of Materials	HI-3110 Evaluation Board PCB Rev-2				
Item	***Qty***	***Description***	***Reference***	***DigiKey***	***Mfr P/N***
1	1	PCB, Bare, Evaluation Board	N/A	--------	
3	1	RS-232 Serial Cable	Separated ordering	AE1379-ND	AK131-2-R
4	4	Capacitor, Cer 10pF 50V 5% NP0 0805	C11,C12,C15,C19	399-1108-1-ND	C0805C100J5GACTU
5	1	Capacitor, Cer 470pF 50V 5% X7R 0805	C22	399-1133-1-ND	C0805C471J5GACTU
6	6	Capacitor, Cer 220nF 10% 50V X7R 0805	C13,C14,C23,C25,C26,C27	399-3491-1-ND	Kemet C0805C224K5RACTU
7	1	1uF 6.3V MLCC	C5	490-4354-1-ND	Murata: LLL219R70J105MA01 (do
8	1	4.7uF 10% 6.3V Low ESL	C6	587-1237-1-ND	JWK212C6475KD-T
9	2	Capacitor, Cer 1nF 20% 50V 7XR 0805	C9,C18	399-1146-1-ND	Kemet C0805C102M5RACTU
10	3	Capacitor, Cer 0.01uF 20% 50V 7XR 0805	C8,C17,C24	399-1160-1-ND	Kemet C0805C103M5RACTU
11	13	Capacitor, Cer 0.1uF 10% 25V X7R 0805	C2,C4,C7,C10,C16,C20,C28,C29,C30,C31, C32,C33,C34	399-1168-1-ND	Kemet C0805C104K3RACTU
12	1	Capacitor 10uF 10% 10V 1206	C3	399-3684-1-ND	Kemet T491A106K010AT
13	2	Capacitor 68uF,20%, 16V Tant SMD	C1,C21	495-2254-1-ND	T491D686M016ZT
14	1	Ferrite Bead	L1	490-5221-1-ND	BLM18PG221SN1D
15	1	LC Filter 2200pF 1206	LCF1 (Do Not Install)	490-2547-1-ND	Murata NFE31PT222
16	2	Connector DB9F, RA PCB, .315"	J2,J10	A35107-ND	Amp 1734354-1
17	1	Res 220K, 1/8W 5% 0805 SMD	R34	RHM220KACT-ND	MCR10EZHJ224
18	2	Header, Male 1x8, 0.1" Pitch	J1,J7 (Do Not Install)		
19	2	Header, Male 2X14, 0.1" Pitch	J5,J6	S2012E-36-ND	Sullins PEC36DAAN
20	0	Header, Male 1x18, 0.1" Pitch	J8 (Do Not Install)		
21	1	Header, Male 0.1", Right Angle 2 x 3	J9	S2312E-36-ND	Sullins PEC36DAAN
22	6	Header, Male 1x2, 0.1" Pitch	JP1-JP5,JP7	S1012E-36-ND	Sullins PEC36SAAN
23	9	LED Green 0805	LED1-LED7,LED10,LED11	160-1179-1-ND	LiteOn LTST-C170GKT
24	1	LED Yellow 0805	LED9	160-1175-1-ND	LiteOn LTST-C170YKT
25	1	LED Red 0805	LED8	160-1176-1-ND	LiteOn LTST-C170CKT
26	1	Res 1M, 1/8W 5% 0805 SMD	R40	RHM1.0MARCT-ND	Panasonic ECG ERJ-6GEYJ105V
27	17	Res 3.3K, 1/8W 5% 0805 SMD	R17,R18,R20-R22,R24-R26,R29-R33,R35-R38	RHM3.3KACT-ND	MCR10EZHJ332
28	11	Res 680, 1/8W 5% 0805 SMD	R1-R6,R8,R11,R12,R14,R16	RHM680ARCT-ND	MCR10EZPJ681
29	2	Res, 4.02 OHM 1% 1/8W 0805	R9,R10	311-4.02CRCT-ND	RC0805FR-074R02L
30	2	Res 60.4 Ohm 1/4W 1% 1210 SMD	R15,R19	RHM60.4BDCT-ND	MCR25JZHF60R4
31	1	Res 0 ohm, 1/8W 5% 0805 SMD	R7	RHM0.0ARCT-ND	MCR10EZPJ000
32	1	Res 1.5M, 1/8W 5% 0805 SMD	R39	RHM1.5MARCT-ND	RHM1.5MARCT-ND
33	1	Res 4.7K, 1/8W 1% 0805 SMD	R28	RHM4.70KCRCT-ND	MCR10EZPF4701
34	1	DIP Switch 4-Pos Slide SMD	SW2	CT2194LPST-ND	CTS 219-4LPST
35	1	DIP Switch 6-Pos Slide SMD	SW1	CT2196LPST-ND	CTS 2196LPST
36	5	Switch Tactile SPST-NO 0.05A 32V	S1,S2,S3,S4,RESET BUTTON	CKN9195CT-ND	KSC222J LFS
37	1	Polyzen 5.6V PPTC Zener SMD	U2	ZEN056V130A24LSCT-ND	Teconn ZEN056V130A24LS
38	2	Test Point, Red Insulator, 0.062" hole	TP1, TP9	5010K-ND	Keystone 5010
39	3	Test Point, Black Insulator, 0.062" hole	TP5,TP6,TP8	5011K-KD	Keystone 5011
40	1	Test Point White Insulator, 0.062" hole	TP2	5012K-ND	Keystone 5012
41	2	Test Point Orange Insulator, 0.062" hole	TP3,TP4	5013K-ND	Keystone 5013
42	1	IC, MC9S12XDT512CAA 80 QFP 16-Bit MCU,	U1	MC9S12XDT512CAA-ND	MC9S12XDT512CAA-ND
43	1	IC, HI-3110 18-SOIC WB PKG	U3	Holt IC	HOLT HI-3110PSI
44	1	IC, MAX3232CSE Narrow 16-SOIC	U4	MAX232CSE+-ND	Texas Inst MAX3232CDR
45	1	Crystal 24MHz, SMD, 50ppm 20pF load	Y1	631-1020-1-ND	FOXSDLF/240F-20
46	1	Crystal 4.00MHz, SMD, 50ppm 20pF load	Y2	631-1005-1-ND	FOXSDLF/040
47	1	OSC 24MHz, 5.0V, 1/2 SIZE	OSC1 (Do Not Install)	XC275-ND	ECS-2200B-240
48	5	3M Bumpon	Install at four corners and center.	SJ5746-0-ND	3M: SJ61A1


MC9S12XDT512CAA
HI-3110 SOIC
MAX232CSE/CD
HI-3110 Demo Board (General Purpose SPI Demo Board) PCB-A
Wednesday, January 26, 2011
BIT RATE SELECT
MODE FUNCTION SELECT
OPTION SELECT
Note: For normal operation SW2-4 should be in the open position.
SMD 6-POS DIP Switch
SW DIP-8_3
CONN HEADER 14x2
HEADER 2X3
DB9F SERIAL PORT
DB9 FEMALE CAN BUS
Chassis grounds very close together with large PCB connection.
HI-3110:
SOLDER JUMPER JP8, JP9
INSTALL JUMPERS: JP3, J2, J5, JP7, JP4