HOLT INTEGRATED CIRCUITS INC.
23351 Madero,
Mission Viejo.
CA 92691. USA.
Tel: + 1 949 859 8800
Fax: + 1 949 859 9643
Email: sales@holtic.com
Web: www.holtic.com

HI-6120 / HI-6121
MIL-STD-1553 Remote Terminal Developer's Kit

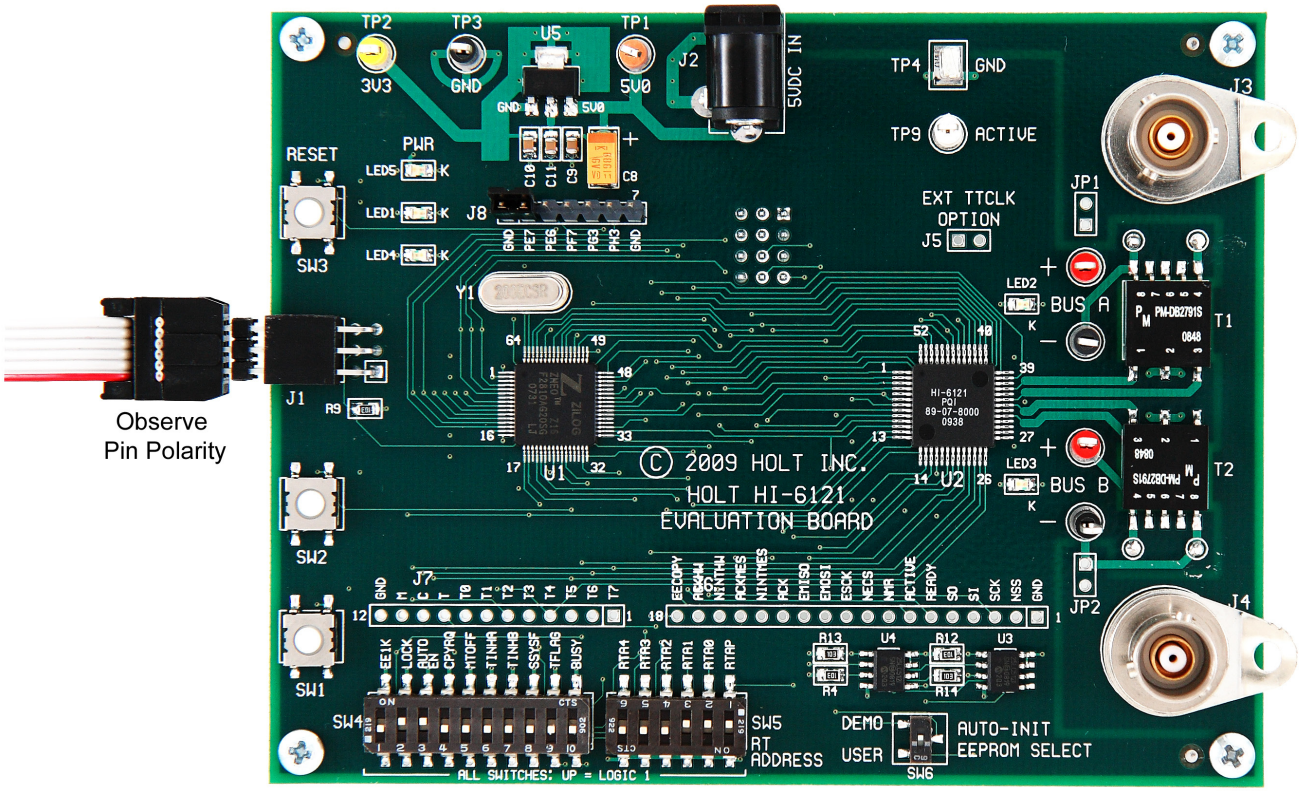
User's Guide

This page intentionally blank

Table of Contents

1. INTRODUCTION

The **HI-6120 / HI-6121 Remote Terminal Developer's Kit** is a reference design for a MIL-STD-1553B Remote Terminal. The **HI-6120 kit** uses a 16-bit parallel bus interface to a Zilog® ZNEO™ microprocessor, while the **HI-6121 kit** uses a 4-wire SPI interface. Each board also contains bus interface transformers, enabling connection to a MIL-STD-1553 bus. The Zilog® ZNEO™ microprocessor can be programmed over a USB using the supplied Zilog® "Smart Cable". The board also includes two external flash EEPROM devices which can be used for automatic initialization.



2. KIT CONTENTS

The **HI-6120 / HI-6121 Remote Terminal Developer's Kit** includes the following hardware:

- Quick Start Guide.
- HI-6120 or HI-6121 Development Board (order each separately).
- Zilog® USB "Smart Cable" with Zilog® driver installation instructions.
- 5V DC external power supply with universal AC mains adapter.
- CD ROM containing additional support documentation and software.

3. MODES OF OPERATION

The kit supports two modes of operation, a quick start **Demonstration Mode** and a **Configuration Mode** for more advanced users.

3.1. Demonstration Mode Set-up

The purpose of the Demonstration Mode is to allow the user to plug into a MIL-STD-1553 bus and respond automatically to valid commands. If the user wishes to make changes to the included software, the board should be set up in Configuration Mode as outlined in Section 3.2.

1. Connect the board to an existing MIL-STD-1553 bus with active bus controller.
2. Set LOCK = “1” (UP position) and AUTOEN = “1” using DIP switch SW4.
3. Set desired RT address bits and correct parity using DIP switch SW5.
4. Select the desired automatic response mode by setting DIP switch SW6. This will auto-initialize the terminal from one of the on-board EEPROM devices.

   Set SW6 to “DEMO” setting: This enables the broadcast command option and Illegal Command Detection[1]. Command Illegalization, interrupts and data buffer assignments are defined in the file `612x_initialization.c`.

   Set SW6 to “USER” setting: This disables the broadcast command option and Illegal Command Detection. In this case the terminal responds “clear status” to all valid commands and ignore invalid commands, including broadcast. Command interrupts and data buffer assignments are defined in the file `612x_initialization.c`.

   NOTE: The “DEMO” setting may also be initialized from the external Zilog® host microprocessor by setting AUTOEN = “0” in step 2 and skipping step 4.
5. Power up the board by connecting the supplied external 5V power supply. If initializing from the host, LED1 will flash green at reset. If initializing from the EEPROM, LED4 will flash red at reset.

3.2. Configuration Mode Set-up

Configuration Mode is intended for advanced users who wish to manually configure the HI-6120 / HI-6121. The kit comes with annotated C files which can easily be edited to custom-configure the HI-6120 / HI-6121 using the on-board Zilog® ZNEO™ microprocessor. The C files may also be used as templates for customer designs using other microprocessors.

1. Go to http://www.zilog.com.
2. Click ***Tools and Software*** → ***Software Downloads***. Download Zilog® Developer Studio II (ZDS II) for ZNEO™ Z16F.
3. Install the ZDS II software onto a Windows computer using default settings.
4. Copy the `Holt HI6120 Demo` (or `Holt HI6121 Demo`) folder and subfolders from the included CD-ROM to the local hard drive. Each C source file in the `Source` directory has a corresponding header (.h) file in the `Include` directory.
5. Open ZDS II. Click ***File*** → ***Open Project,*** navigate to the `Source` subfolder, select project file `Zneo_6120_demo.zdsproj` and open it.

1 The following commands will be illegalized: Undefined Mode Codes, Reserved Mode Codes, Broadcast Transmit Subaddress Commands, Broadcast Transmit Mode Codes 0,2,16,18,19, Transmit Mode Codes 0,3 and all Transmit Subaddress Commands for Subaddress 17.

6. Click ***Rebuild All*** and the project should compile without errors or warnings.

7. Set LOCK = “0” (DOWN position), CPYREQ = “0” and AUTOEN = “0” using DIP switch SW4.

8. Set desired RT address bits and correct parity using DIP switch SW5.

9. Connect Zilog® Smart Cable to USB port. This “plug and play” device should be recognized by Windows and declared ready to use. Apply power to the HI-612x demo board. Observing the ribbon cable pin 1 polarity (stripe shown in the photo), plug the Smart Cable ribbon cable into the board using the supplied male-male adapter.

10. Click ***Debug*** → ***Reset*** and the Zilog® debugger should successfully “Connect to Target” and download the program into the processor flash memory. The window showing `main.c` should show a yellow program counter arrow at the top of `main.c`.

4. SOFTWARE FILES

The following list summarizes the main files which may be edited to change the configuration options of the HI-6120 / HI-6121. Also listed are files required to initialize the Zilog® ZNEO™ processor.

`Holt HI6120 Demo` (or `Holt HI6121 Demo`) folder

`Include` folder

```
612x_initialization.h
612x_test.h
bb_spi.h
bus_interface.h
espi.h
global_definition.h
gpio.h
system_clock.h
timer.h
```

`Source` folder

```
612x_initialization.c
612x_test.c
bb_spi.c
bus_interface.c
espi.c
gpio.c
main.c
system_clock.c
timer.c
Zneo_6120_demo.zdsproj
```

4.1. 612x_initialization.h

This header file defines top level functionality.

- Define which host bus interface is used, HI-6120 (16-bit parallel) or HI-6121 (SPI):

```
#define HOST_BUS_INTERFACE     NO     // YES = HI-6120 host bus interface
                                      // NO = HI-6121 host SPI interface
```

- Enable / Disable Illegal Command Detection

```
#define ILLEGAL_CMD_DETECT      YES    // YES = Enable Illegal Command Detection
                                       // NO = Disable Illegal Command Detection
                                       // "in form" response for all valid commands
```

- Enable / Disable Broadcast Commands valid

```
#define SUPPORT_BROADCAST       YES    // YES = commands to RT31 are valid
                                       // NO = commands to RT31 are invalid, ignored
```

- Segregate Broadcast Commands for Notice 2 compliance

```
*\brief     Macros for HI-612X Configuration Register 1
#define NOTICE2          0x0004      // notice 2 separate broadcast data
```

- Enable / Disable Undefined Mode Codes valid

```
#define UNDEF_MCODES_VALID      YES    // YES = undefined & reserved mode commands
                                       // are valid
                                       // NO = undefined & reserved mode commands
                                       // are invalid, ignored
```

- Enable / Disable Simplified Mode Command Processing

```
#define USE_SMCP    YES    // YES = mode command results stored in descriptor
                           // table itself, not in assigned RAM buffers
                           // NO = descriptor table is initialized to store
                           // mode command results in assigned RAM buffers
```

4.2. bus_interface.h (HI-6120 only)

This file may be used to configure the host bus interface for HI-6120. This file should be modified in conjunction with a number of jumper settings (see schematic).

- Set "Intel" or "Motorola" style bus control using the BTYPE JP8 jumper

  OPEN = "Intel" Style ($\overline{WE}$ & $\overline{OE}$)
  CLOSED = "Motorola" Style (R/$\overline{W}$ & $\overline{STR}$)

- Define bus width

```
#define BUS_8_BITS              NO     /*!< default is 16-bit bus width */
                                YES    /*!< select 8-bit bus width */
```

  If 8-bit is selected, the BWID jumper JP9 must be closed
  OPEN = 16-bit wide bus
  CLOSED = 8-bit wide bus,

  and the BENDI jumper JP6 may be used to select endianness
  OPEN = Big Endian byte order
  CLOSED = Little Endian byte order.

- Enable / Disable Host bus read cycle 'wait'. A WAIT output may be enabled when the host processor read cycle time is too fast for the worst case (slowest) read cycle time of HI-6120.

```
#define BUS_WAIT_INPUT_ENABLED NO     /*!<  YES enables the
                                            ZNEO™ bus WAIT input pin      */
```

The WAIT polarity is set by the WPOL JP7 jumper

OPEN	=	Active High WAIT
CLOSED	=	Active Low WAIT.

5. HI-6120 / HI-6121 DEVELOPER'S KIT MEMORY MAP

The following assumptions are made in describing the Developer's Kit memory map:

1. All addresses shown are expressed as hexadecimal values.
2. Addressing for HI-6121 uses device internal addresses. Bus addressing for HI-6120 is offset by chip select base address 0x00800000 and the microprocessor uses byte addressing so all address offsets are doubled. (The LSB becomes upper/lower byte select for each word.)
3. Memory allocations are shared for undefined and reserved mode code commands, and unimplemented sub-address commands. These commands are grouped by like requirements, and share common RAM resources (bit bucket).
4. For messages needing an odd number of words, an extra "pad" word is added so the next buffer begins at an even address.
5. Subaddresses using circular buffer Mode 1 are followed by a 32-word overrun buffer, in case a 32 data word receive command arrives with just one location remaining before "buffer full" attainment.

5.1. Descriptor Table Partitioning

Descriptor Table Sector	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)	
	Start	End	Start	End
Receive Subaddress	0200	027F	800400	8004FE
Transmit Subaddress	0280	02FF	800500	8005FE
Receive Mode Codes	0300	037F	800600	8006FE
Transmit Mode Codes	0380	03FF	800700	8007FE

5.2. Buffer Assignments for Receive and Transmit Subaddresses

Receive (Rx) Subbaddress or Transmit (Tx) Subaddress	Buffer Method and Data Pointer(s)		Buffer Size (Words)	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)		Structures Reserved *MIW = Msg Info Word TT = Time Tag Word*
				Start	*End*	*Start*	*End*	
Rx SA1 (data pointers A, B and broadcast data pointer)	ping-pong	DPA DPB BDP	34 34 34	0400 0422 0444	0421 0443 0465	800800 800844 800888	800842 800886 8008CA	MIW + TT + 32 words same same
Tx SA1 (data pointers A, B and broadcast data pointer)	ping-pong	DPA DPB BDP	34 34 4	0466 0488 04AA	0487 04A9 04AD	8008CC 800910 800954	80090E 800952 80095A	same same MIW + TT + 2 pad
Rx SA30 and Tx SA30 for data wrap-around	index-0	DPA	34	04AE	04CF	80095C	80099E	MIW + TT + 32 words

Receive (Rx) Subbaddress or Transmit (Tx) Subaddress	Buffer Method and Data Pointer(s)		Buffer Size (Words)	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)		Structures Reserved *MIW = Msg Info Word* *TT = Time Tag Word*
				Start	*End*	*Start*	*End*	
Rx SA2	index-32	DPA BDP	1088 34	04D0 0910	090F 0931	8009A0 801220	80121E 801262	32 x (MIW + TT + 32 words)
Tx SA2	index-32	DPA BDP	1088 4	0932 0D72	0D71 0D75	801264 801AE4	801AE2 801AEA	MIW + TT + 2 pad
Rx SA3	circ1-32	DPA pad	1088 32	0D76 11B6	11B5 11D5	801AEC 80236C	80236A 8023AA	32 x (MIW + TT + 32 words) pad for overrun
Tx SA3	circ1-32	DPA pad	1088 32	11D6 1616	1615 1635	8023AC 802C2C	802C2A 802C6A	32 x (MIW + TT + 32 words) pad for overrun
shared buffer: all unimplemented Rx subaddresses	index-0	DPA	34	1636	1657	802C6C	802CAE	MIW + TT + 32 words
shared buffer: all unimplemented Tx subaddresses	index-0	DPA	34	1658	1679	802CB0	802CF2	MIW + TT + 32 words
unassigned RAM	---	---	390	167A	17FF	802CF4	802FFE	-
Rx & Tx SA4	circ-2 256 msg max	MIB DPA	512 8192	1800 1A00	19FF 39FF	803000 803400	8033FE 8073FE	256 x (MIW + TT)
unassigned RAM	---	---	2560	3A00	43FF	807400	8087FE	-

5.3. Shared Buffer Assignments for Undefined and Reserved Mode Code Commands

NOTE: *These RAM buffer allocations for mode code commands only apply when the application does not use the SMCP option.*

Undefined & Reserved Receive (Rx) Mode Codes Transmit (Tx) Mode Codes	Buffer Method and Data Pointer(s)		Buffer Size (Words)	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)		Structures Reserved *MIW = Msg Info Word* *TT = Time Tag Word*
				Start	*End*	*Start*	*End*	
shared buffer: undefined Rx MC0 - MC15	index-0	DPA	4	0060	0063	8000C0	8000C6	MIW + TT, 0 data, 2 pad
shared buffer: undefined Rx MC16, undefined Rx MC18 - MC19, reserved Rx MC22 - MC31	index-0	DPA	4	0064	0067	8000C8	8000CE	MIW + TT, 1 data, 1 pad
shared buffer: undefined Tx MC9 - MC15	index-0	DPA	4	0068	006B	8000D0	8000D6	MIW + TT, 0 data, 2 pad
shared buffer: undefined Tx MC17, undefined Tx MC20 - MC21, reserved Tx MC22 - MC31	index-0	DPA	4	006C	006F	8000D8	8000DE	MIW + TT, 1 data, 1 pad

5.4. Buffer Assignments for Defined Transmit Mode Code Commands MC0 - MC8 (No Data Word)

NOTE: *These RAM buffer allocations for mode code commands only apply when the application does not use the SMCP option.*

Defined Transmit (Tx) Mode Code Commands, No Data	Buffer Method and Data Pointer(s)		Buffer Size (Words)	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)		Structures Reserved *MIW = Msg Info Word TT = Time Tag Word*
				Start	*End*	*Start*	*End*	
Tx MC0	ping-pong	DPA DPB BDP	2 2 2	0070 0072 0074	0071 0073 0075	8000E0 8000E4 8000E8	8000E2 8000E6 8000EA	MIW + TT same same
Tx MC1	ping-pong	DPA DPB BDP	2 2 2	0076 0078 007A	0077 0079 007B	8000EC 8000F0 8000F4	8000EE 8000F2 8000F6	MIW + TT same same
Tx MC2	ping-pong	DPA DPB BDP	2 2 2	007C 007E 0080	007D 007F 0081	8000F8 8000FC 800100	8000FA 8000FE 800102	MIW + TT same same
Tx MC3	ping-pong	DPA DPB BDP	2 2 2	0082 0084 0086	0083 0085 0087	800104 800108 80010C	800106 80010A 80010E	MIW + TT same same
Tx MC4	ping-pong	DPA DPB BDP	2 2 2	0088 008A 008C	0089 008B 008D	800110 800114 800118	800112 800116 80011A	MIW + TT same same
Tx MC5	ping-pong	DPA DPB BDP	2 2 2	008E 0090 0092	008F 0091 0093	80011C 800120 800124	80011E 800122 800126	MIW + TT same same
Tx MC6	ping-pong	DPA DPB BDP	2 2 2	0094 0096 0098	0095 0097 0099	800128 80012C 800130	80012A 80012E 800132	MIW + TT same same
Tx MC7	ping-pong	DPA DPB BDP	2 2 2	009A 009C 009E	009B 009D 009F	800134 800138 80013C	800136 80013A 80013E	MIW + TT same same
Tx MC8	ping-pong	DPA DPB BDP	2 2 2	00A0 00A2 00A4	00A1 00A3 00A5	800140 800144 800148	800142 800146 80014A	MIW + TT same same

5.5. Buffer Assignments for Defined Transmit Mode Code Commands MC16, MC18 and MC19 (1 Data Word)

NOTE: *These RAM buffer allocations for mode code commands only apply when the application does not use the SMCP option.*

Defined Transmit (Tx) Mode Code Commands with Data Word	Buffer Method and Data Pointer(s)		Buffer Size (Words)	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)		Structures Reserved *MIW = Msg Info Word TT = Time Tag Word*
				Start	*End*	*Start*	*End*	
Tx MC16	ping-pong	DPA DPB BDP	4 4 4	00A6 00AA 00AE	00A9 00AD 00B1	80014C 800154 80015C	800152 80015A 800162	MIW + TT, 1 data, 1 pad same same
Tx MC18	ping-pong	DPA DPB BDP	4 4 4	00B2 00B6 00BA	00B5 00B9 00BD	800164 80016C 800174	80016A 800172 80017A	MIW + TT, 1 data, 1 pad same same
Tx MC19	ping-pong	DPA DPB BDP	4 4 4	00BE 00C2 00C6	00C1 00C5 00C9	80017C 800184 80018C	800182 80018A 800192	MIW + TT, 1 data, 1 pad same same

5.6. Buffer Assignments for Defined Receive Mode Code Commands MC17, MC20 and MC21 (1 Data Word)

NOTE: *These RAM buffer allocations for mode code commands only apply when the application does not use the SMCP option.*

Defined Transmit (Tx) Mode Code Commands with Data Word	Buffer Method and Data Pointer(s)		Buffer Size (Words)	Device Internal Addr same as HI-6121 Addr		Data Bus Addr Hex (HI-6120 only)		Structures Reserved *MIW = Msg Info Word TT = Time Tag Word*
				Start	*End*	*Start*	*End*	
Rx MC17	ping-pong	DPA DPB BDP	4 4 4	00CA 00CE 00D2	00CD 00D1 00D5	800194 80019C 8001A4	80019A 8001A2 8001AA	MIW + TT, 1 data, 1 pad same same
Rx MC20	ping-pong	DPA DPB BDP	4 4 4	00D6 00DA 00DE	00D9 00DD 00E1	8001AC 8001B4 8001BC	8001B2 8001BA 8001C2	MIW + TT, 1 data, 1 pad same same
Rx MC21	ping-pong	DPA DPB BDP	4 4 4	00E2 00E6 00EA	00E5 00E9 00ED	8001C4 8001CC 8001D4	8001CA 8001D2 8001DA	MIW + TT, 1 data, 1 pad same same
unassigned RAM	-	-	18	00EE	00FF	8001DC	8001FE	Do NOT exceed 0x00FF !

6. ZDS II DEBUGGER “MEMORY WATCH” WINDOW

The Zilog® Developer Studio II (ZDS II) for ZNEO™ provides a powerful debugging tool for viewing memory address space, the “Memory Watch” window. This tool is accessed in the debugger by clicking ***View → Debug Windows → Memory***. This tool can display any selected range of register or RAM memory address space in the HI-6120. It is a useful element when debugging HI-6120 applications, providing the ability to easily view changes to registers and memory occurring as a result of host write operations or message processing. The “Memory Watch” window is updated whenever program execution stops, e.g. due to a debug breakpoint or clicking the Stop button in the ZDS II debugging window.

When selecting the displayed memory address space, the user must be mindful of the addressing rules, which differ for the HI-6120 and HI-1621.

6.1. HI-6120 addressing

The HI-6120 is memory mapped within the ZNEO™ processor’s memory address range. The processor has a 24-bit memory address space. Using the ZNEO™ version with external bus interface, the HI-6120 32K word address range appears at the processor bus address corresponding to the hardware chip select that enables the HI-6120. In the accompanying evaluation board, the HI-6120 chip select has a start address of 0x0080000. This corresponds to HI-6120 register address 0x0000, Configuration Register 1. Thus the C program reads and writes this register at bus address 0x800000.

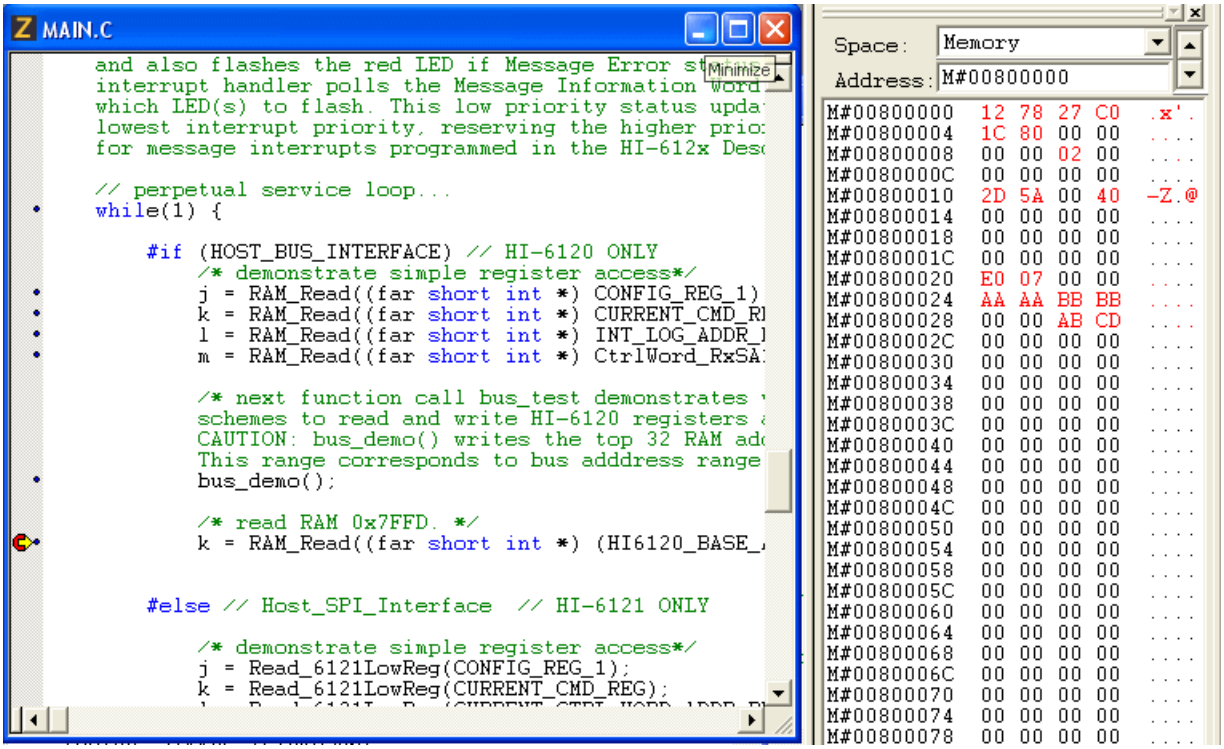
The ZNEO™ processor external bus uses byte addressing while the Holt C program uses word addressing. For word addressing, all memory accesses must use even addresses for proper alignment. The C program uses a simple scheme for addressing the HI-6120: the desired register/RAM address is left-shifted and added to the chip select offset, 0x800000:

HI-6120 Reg/RAM Address	Conversion to Bus Address (0x800000 + (addr << 1))	Bus Address
0x0000	0x800000 + (0x0000 << 1))	0x800000
0x0001	0x800000 + (0x0001 << 1))	0x800002
0x001A	0x800000 + (0x001A << 1))	0x800034
0x03D5	0x800000 + (0x03D5 << 1))	0x8007AA
0x7FFF	0x800000 + (0x7FFF << 1))	0x80FFFE (last RAM addr)

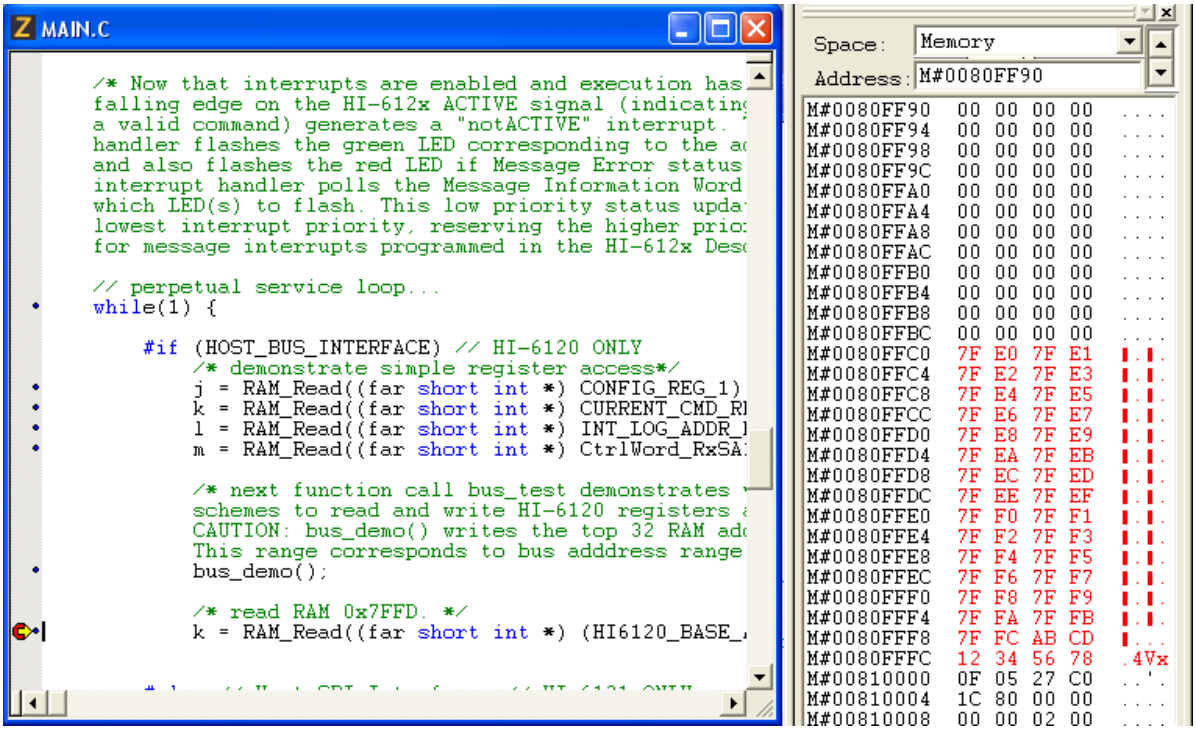
The C program uses the “Conversion to Bus Address” format for specifying read/write locations, so the desired HI-6120 internal address is evident to the programmer. However, the ZDS II “Memory Watch” window when debugging uses the “Bus Address,” so it may be necessary from time to time to convert displayed bus address back to the internal word address in the HI-6120.

The Memory Watch window can be resized to display any number of bytes per row. To avoid confusion, only use an even number of bytes per row. In the screen capture, the window is sized for 4 bytes (two 16-bit words) per row.

When program execution stops and a Memory Watch window is open, the debugger refreshes the displayed memory range, reading each location and updating the values shown in the window. The screen capture below shows a segment of the Memory Watch debug window. Here, the Memory Watch start address is the HI-6120 register 0 address, occurring at processor bus address 0x800000. This is the chip select start address for the HI-6120. The valid bus address range for viewing HI-6120 data is 0x800000 through 0x80FFFF. The values displayed in red are the initialized register values, after reset. There are 2 words per row, so registers 0 and 1 contain, respectively 0x1278 and 0x27C0 as shown in the example screen capture below.



The screen capture below shows the bottom of the 32K word address range in the HI-6120. The last RAM word is at bus address 0x80FFFE, corresponding to internal HI-6120 address 0x7FFF. In this capture, the last 32 words in RAM were modified by the function call bus_demo( ) executed just before the breakpoint that stopped execution. All but the last 3 words contain the internal offset address within the HI-6120 device 0x7FE0 through 0x7FFC.



Be mindful that reading certain registers and RAM locations (including refresh for the Memory Watch window) changes the stored value in the device and may alter program execution after resuming:

- Reading the Pending Interrupt register automatically resets the register to zero after reading.
- Reading any Control Word in the Descriptor Table automatically resets its DBAC “data block accessed” bit. If this bit is used for polling Control Words to detect message activity (one of several message detection options), the bit will be reset after the Memory Watch window is refreshed.

The Memory Watch window is a powerful tool for viewing the effects of MIL-STD-1553 command processing on the addressed data buffers.

6.2. HI-6121 addressing

The SPI interface between host CPU and HI-6121 does not permit direct access to view register or RAM data using the Zilog® ZDS II Memory Watch window. The Holt demo program for the HI-6121 provides the ability to view a selected 256 word range within the register/RAM address space.

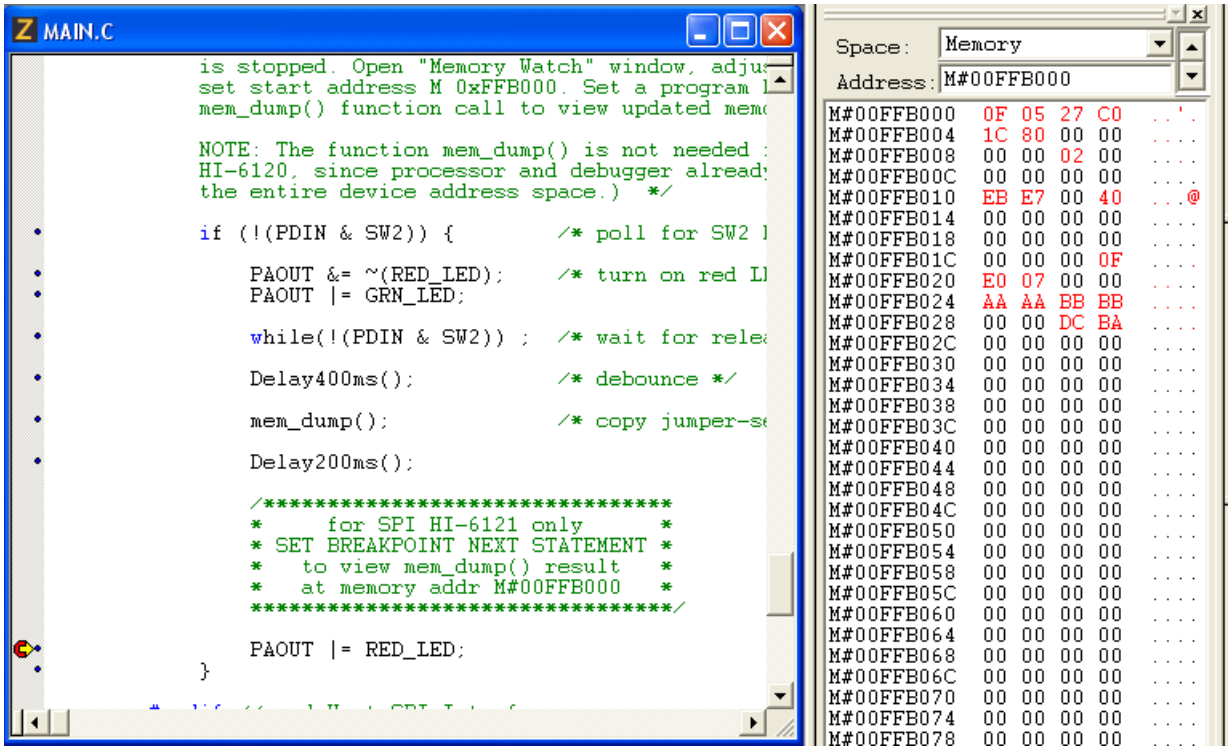
In the example C program, set to compile for HI-6121, the primary function main( ) polls the evaluation board pushbutton SW2. If SW2 is pressed, the program copies a jumper-selected range of 256 words from the HI-6121, into a fixed memory address range within internal ZNEO™ processor RAM. Once copied there, the selected data can be displayed using the Memory Watch window. To view the copied data, the debug window should have an execution breakpoint set after the routine detects SW2 closure and has copied HI-6121 data to internal processor RAM. The suggested breakpoint location is clearly indicated in the main( ) source file.

When debugging the HI-6121 with the Memory Watch window, the start address for viewing will always be “M 0x0FFFB000”, the start address for internal ZNEO™ RAM. Before pressing button SW2, the user specifies the desired 256-word range by setting a jumper on the board’s header J8. There are 7 different memory start addresses based on jumper position, including “no jumper”. The pre-assigned data block start addresses are easily reprogrammed to support application debugging situations.

The Memory Watch window will always have a start address of “M 0x0FFFB000” when debugging the HI-6121. The user will have to convert displayed memory address into HI-6121 internal address. Unlike HI-6120 debug, displayed Memory Watch addresses cannot be directly converted into corresponding HI-6121 internal addresses. However, the displayed address can be converted to a word offset from the start of the selected 256-word data block in the HI-6121:

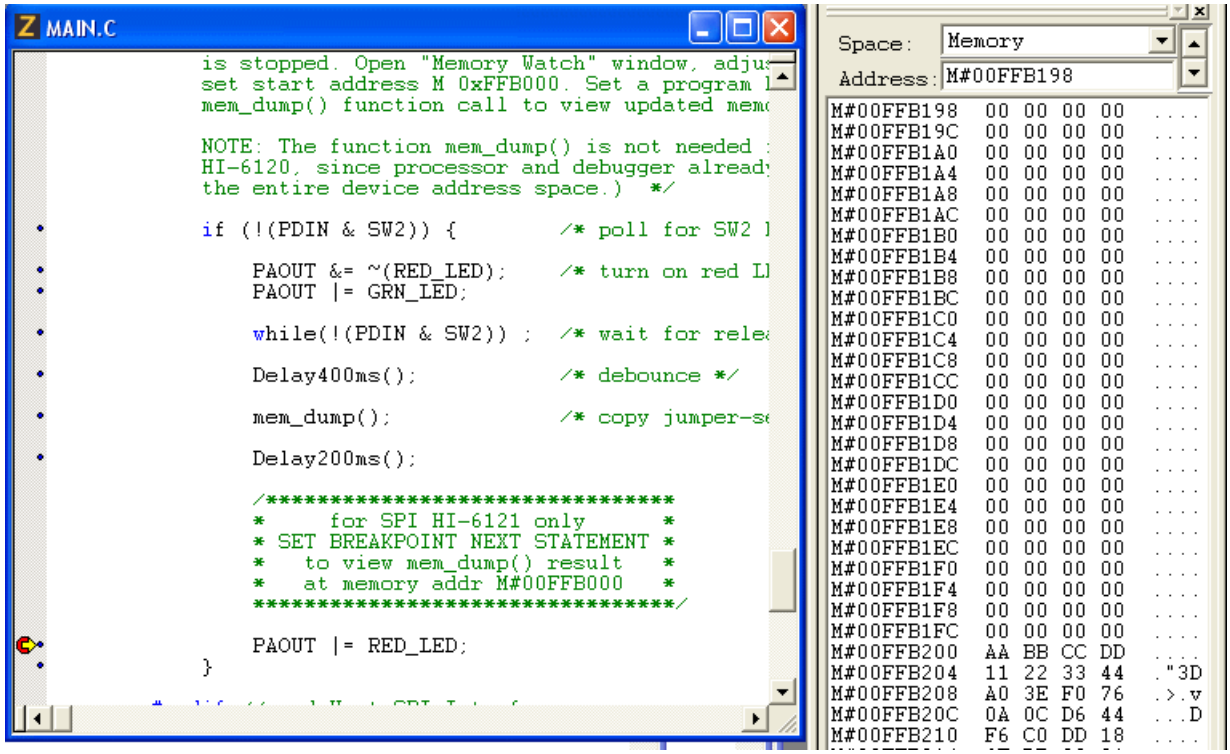
```
Bus          Convert to HI-6121 offset (word)      word       word
Address      (bus_address - 0x0FFFB000) >> 1       offset     number

0x0FFFB000   ((0x0FFFB000 - 0x0FFFB000) >> 1)  =  0x00       0
0x0FFFB002   ((0x0FFFB002 - 0x0FFFB000) >> 1)  =  0x01       1
0x0FFFB034   ((0x0FFFB034 - 0x0FFFB000) >> 1)  =  0x1A       26
0x0FFFB0FF   ((0x0FFFB0FF - 0x0FFFB000) >> 1)  =  0x7F       127
0x0FFFB1FE   ((0x0FFFB1FE - 0x0FFFB000) >> 1)  =  0xFF       255
```

The screen capture below shows a segment of the Memory Watch debug window, using the scheme developed for HI-6121. The Memory Watch start address is the fixed start-of RAM address in the ZNEO™ processor, 0x00FFB000. The values displayed in red are the initialized register values, after reset. There are 2 words per row. The code window shows the breakpoint location below function call mem_dump( ), occurring only after button SW2 is pressed:



The screen capture below shows the bottom of the 256-word data block captured by the mem_dump( ) function. The last word address is 0xFFB1FE in row 0x00FFB1FC. The next two rows contain a visual demarcation for end-of-buffer, containing:

```
0x AA BB CC DD
0x 11 22 33 44
```



7. HI-6121 — INTERRUPTS DURING MULTI-WORD SPI BYTE TRANSFERS

Multi-word SPI transfers between the host and HI-6121 generally have three phases:

a. The host initializes the Memory Address Pointer (register 15) by direct write or by using certain SPI op codes that modify or load register 15

b. The host sends the op code byte to HI-6121 SPI to commence N-word SPI transfer,

c. N 16-bit words are transacted between the host and HI-6121, relying on auto-increment of the Memory Address Pointer between words.

Problems may arise when interrupts occur during an SPI sequence. The interrupt service routine (ISR) seizes SPI control to determine the interrupt source and take action. Upon completion of the interrupt service routine, program execution returns to the interrupted procedure, at the point where the interrupt was recognized. The auto-increment aspect of Memory Address Pointer (register 15) was cancelled when SPI control was seized by the interrupt. Further, the address pointer value no longer applies to the interrupted process, unless the interrupt service routine only used directly-addressed SPI op codes to read registers 0-15. These op codes do not alter register 15 contents.

Careful software design can prevent SPI disruption when interrupts occur. Alternative strategies are described below.

7.1. Alternative Interrupt Management Strategies

1. Use polled interrupts rather than vectored interrupts. Configure HI-612x interrupt outputs as “level outputs” not “pulse outputs” (see datasheet for details). Only read interrupt status at controlled points in program execution, to avoid overlapping demand on SPI interface.

2. Use vectored interrupts, but globally disable interrupts whenever SPI is busy. This approach delays recognition of new interrupt requests occurring while SPI is active running foreground tasks, and may add considerable latency for interrupt servicing if SPI is transacting large blocks of data while interrupts remain continuously disabled.

3. The programmed foreground SPI tasks disable interrupts during individual byte/word transfers, but re-enable interrupts between words to recognize new interrupt request events. With careful program design, even multi-word foreground spi transfers can recover from interrupts that use SPI. An example follows:

Software structure is shown for main( ), an N-word SPI transfer function called by main( ) and the ISR (interrupt service routine) for a generic vectored interrupt that also uses SPI transfers. Two global variables are introduced, spi_busy and spi_irq …

```
RESET

main( )  {
   system initialization
   spi_busy = 0
   spi_irq = 0


   /* perpetual service loop */
   while(1) {
      housekeeping ;
      periodic functions ;
      /* example function using spi */
      n_word_spi_transfer( )
   } // end while
}   // end main()
```

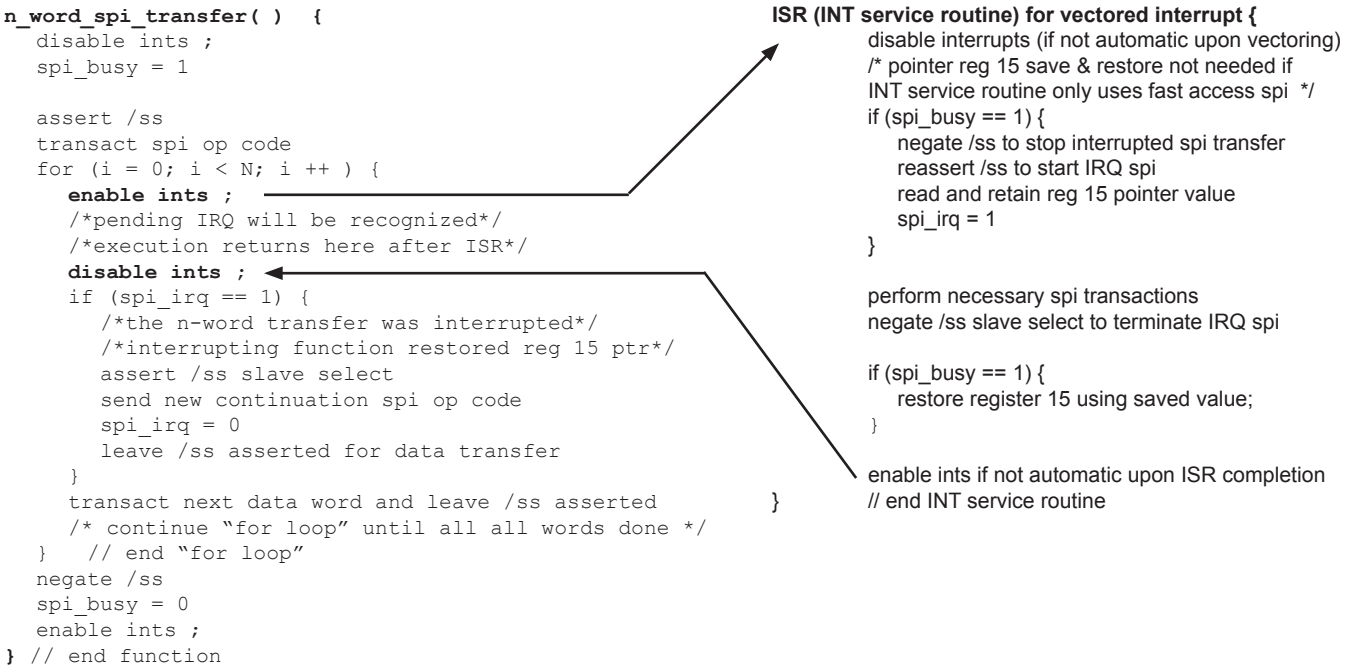
Assumptions:

1. IRQs have priority over foreground periodic activities

2. All vectored IRQs that use SPI have same interrupt priority; an IRQ using SPI cannot be interrupted another IRQ.

Note:

1. IRQs that only use “fast access” SPI op codes (not using register 15) do not have to save/restore reg 15 but the spi_irq flag assertion is essential.


n_word_spi_transfer( ) {
disable ints ;
spi_busy = 1
assert /ss
transact spi op code
for (i = 0; i < N; i ++) {
enable ints ;
/*pending IRQ will be recognized*/
/*execution returns here after ISR*/
disable ints ;
if (spi_irq == 1) {
/*the n-word transfer was interrupted*/
/*interrupting function restored reg 15 ptr*/
assert /ss slave select
send new continuation spi op code
spi_irq = 0
leave /ss asserted for data transfer
}
transact next data word and leave /ss asserted
/* continue "for loop" until all all words done */
} // end "for loop"
negate /ss
spi_busy = 0
enable ints ;
} // end function
ISR (INT service routine) for vectored interrupt {
disable interrupts (if not automatic upon vectoring)
/* pointer reg 15 save & restore not needed if
INT service routine only uses fast access spi */
if (spi_busy == 1) {
negate /ss to stop interrupted spi transfer
reassert /ss to start IRQ spi
read and retain reg 15 pointer value
spi_irq = 1
}
perform necessary spi transactions
negate /ss slave select to terminate IRQ spi
if (spi_busy == 1) {
restore register 15 using saved value;
}
enable ints if not automatic upon ISR completion
}
// end INT service routine

8. BILL OF MATERIALS: HI-6120 DEVELOPMENT KIT

Item	*Qty*	*Description*	*Reference*	*DigiKey*	*MFR P/N*
1	1	PCB, Bare, Evaluation Board	N/A	--------	
2	1	Capacitor, Ceramic 10nF 10% 50V X7R 0805	C1	399-1158-1-ND	Kemet C0805C103K5RACTU
3	2	Capacitor, Ceramic 22pF 5% NPO 50V 0805	C2,C3	311-1103-1-ND	Yaego 0805CG220J9B200
4	2	Capacitor, Ceramic 1uF 10% 16V X7R 0805	C4,C10	399-1284-1-ND	Kemet C0805C105K4RACTU
5	19	Capacitor, Ceramic 0.1uF 20% 50V Z5U 0805	C5, C9, C11 - C27	399-1176-1-ND	Kemet C0805C104M5UACTU
6	3	Capacitor 68uF 10% 6.3V Tantalum SMD EIA 6032-28	C6,C7,C8	495-1507-1-ND	Kemet B45197A1686K309
7	0	Header, Male 2x18, 0.1" Pitch	J6	S2012-18-ND DO NOT STUFF	Sullins
8	0	Header, Male 1x12, 0.1" Pitch	J7	S1012-12-ND DO NOT STUFF	Sullins
9	0	Header, Male 1x18, 0.1" Pitch	J8	S1012-18-ND DO NOT STUFF	Sullins
10	0	Header, Male 1x7, 0.1" Pitch	J10	DO NOT STUFF	Sullins
11	1	Receptacle, 0.1" Pitch, Right Angle 2x3	J1	S5518-ND	Sullins PPTC042LJBN-RC
12	1	Header, Male 2x3, 0.1" Pitch, 0.310" pins one side, 0.320" pins other side	J1	Programming Adapter for J1 to Zilog® Debug Cable	Samtec TSW-103-16-T-D
13	1	Jack, DC Power, 2.5mm ID x 2.1mm pin	J2	CP-102AH-ND	CUI PJ-102AH
14	2	Connector 3-Lug Concentric Triax Bayonet Jack, Panel Front Mount TRB	J3, J4	MilesTek 10-06570	Trompeter Electronics BJ77 Use 0.469 Round Hole
15	0	Header, Male, 0.1" Pitch, 1x2	J5	DO NOT STUFF	
16	0	Header, Male 1x2 0.1" Pitch, 0.230" Pins, 0.120" Tails	JP1,JP2	DO NOT STUFF	Sullins
17	0	Solder Jumper	JP4-JP11	DO NOT SOLDER	
18	1	Header, Male, 0.1” pitch, 3x4	JP12	Samtec	Samtec TSW-104-07-T-T
19	1	Shunt, 4-parallel, 0.1” pitch	JP12	Samtec	Samtec MNT-104-BK-T
20	4	LED Green 0805	LED1 - LED3, LED5	160-1179-2-ND	LiteOn LTST-C170GKT
21	1	LED Yellow 0805	LED6	160-1175-2-ND	LiteOn LTST-C170YKT
22	1	LED Red 0805	LED4	160-1176-2-ND	LiteOn LTST-C170CKT
23	1	Oscillator, SMD 5 x 7 mm Program for 50.0MHz, 3.3V	OSC1	CSX-750PBBCT-ND	Citizen CSX-750P1P-UT, Program for 50.0 MHz

Item	*Qty*	*Description*	*Reference*	*DigiKey*	*MFR P/N*
24	1	Resistor, 3.3 Ohm 5% 1/8W 0805	R18	311-3.3ARCT-ND	Any
25	6	Resistor, 68 5% 1/8W 0805	R1,R2,R3,R5, R15, R19	311-68ARCT-ND	Any
26	7	Resistor, 10K 5% 1/8W 0805	R4,R9,R10,R11 R12,R13,R14	311-10KARCT-ND	Any
27	5	Resistor, 47K 5% 1/8W 0805	R6,R7,R8, R16,R17	311-47KARCT-ND	Any
28	3	Pushbutton	SW1,SW2,SW3	P10886SCT-ND	Panasonic EVQ-QWS02W
29	1	DIP Switch 10-Pos Slide SMD	SW4	CT1910LPST-ND	CTS 219-10LPST
30	1	DIP Switch 6-Pos Slide SMD	SW5	CT2196LPST-ND	CTS 219-6LPST
31	1	Slide Switch SPDT SMD	SW6	563-1022-1-ND	Copal CJS-1200TB
32	2	Transformer MIL-STD-1553 Single Bus 1:2.50 ratio	T1,T2	Holt PM-DB2791S	
33	2	Test Point, Red Insulator, 0.062" hole	TP5,TP7	5010K-ND	Keystone 5010
34	3	Test Point, Black Insulator, 0.062" hole	TP3,TP6,TP8	5011K-KD	Keystone 5011
35	1	Test Point, Orange Insulator, 0.062" hole	TP1	5008K-ND	Keystone 5008
36	1	Test Point, Yellow Insulator, 0.062" hole	TP2	5009K-ND	Keystone 5009
37	2	Test Point, White Insulator, 0.062" hole	TP8,TP9	5012K-ND	Keystone 5012
38	1	Test Point, Compact SMT	TP4	5016KCT-ND	Keystone 5016
39		Test Point, Hole / Pad Only	TP10 -TP15		
40	1	IC, Z16F2811 16-Bit MCU, 128K Flash 0-70C 100-PQFP	U1	Mouser 692-Z16F2811AL20SG	Zilog® Z16F2811AL20SG
41	1	IC, HI-6120 100-PQFP	U2	Holt HI-6120PQI	
42	2	IC, Serial EEPROM 512Kbit 20MHz SPI 8-SOIC, Microchip	U3, U4	25LC512T-I/SNCT-ND	Microchip 25LC512T-I/SN
43	1	IC, 2-In AND Gate SOT23-5	U6	296-11601-1-ND	TI SN74LVC1G08DBVR
44	1	IC Voltage Regulator 3.3V 1A LDO, SOT-223	U5	497-1228-1-ND	ST Micro LD1117AS33TR
45	2	Pin Socket .015 to .021" pin, Use 0.040" hole	Y1	ED5037-ND	MillMax 0552-1-15-01-1127100
46	1	Crystal 20.00MHz, 50ppm 20pF, HC-49US leaded	Y1	631-1111-ND	Fox FOXSLF/S200-20
47	4	Stand-off, #4-40 Female Thread, 1" long	-----	Any	
48	4	Machine Screw, #4-40 x 1/4"	-----	Any	
49	4	Lock Washer, Int.Tooth #4-40	-----	Any	

9. SCHEMATICS: HI-6120 DEVELOPMENT KIT

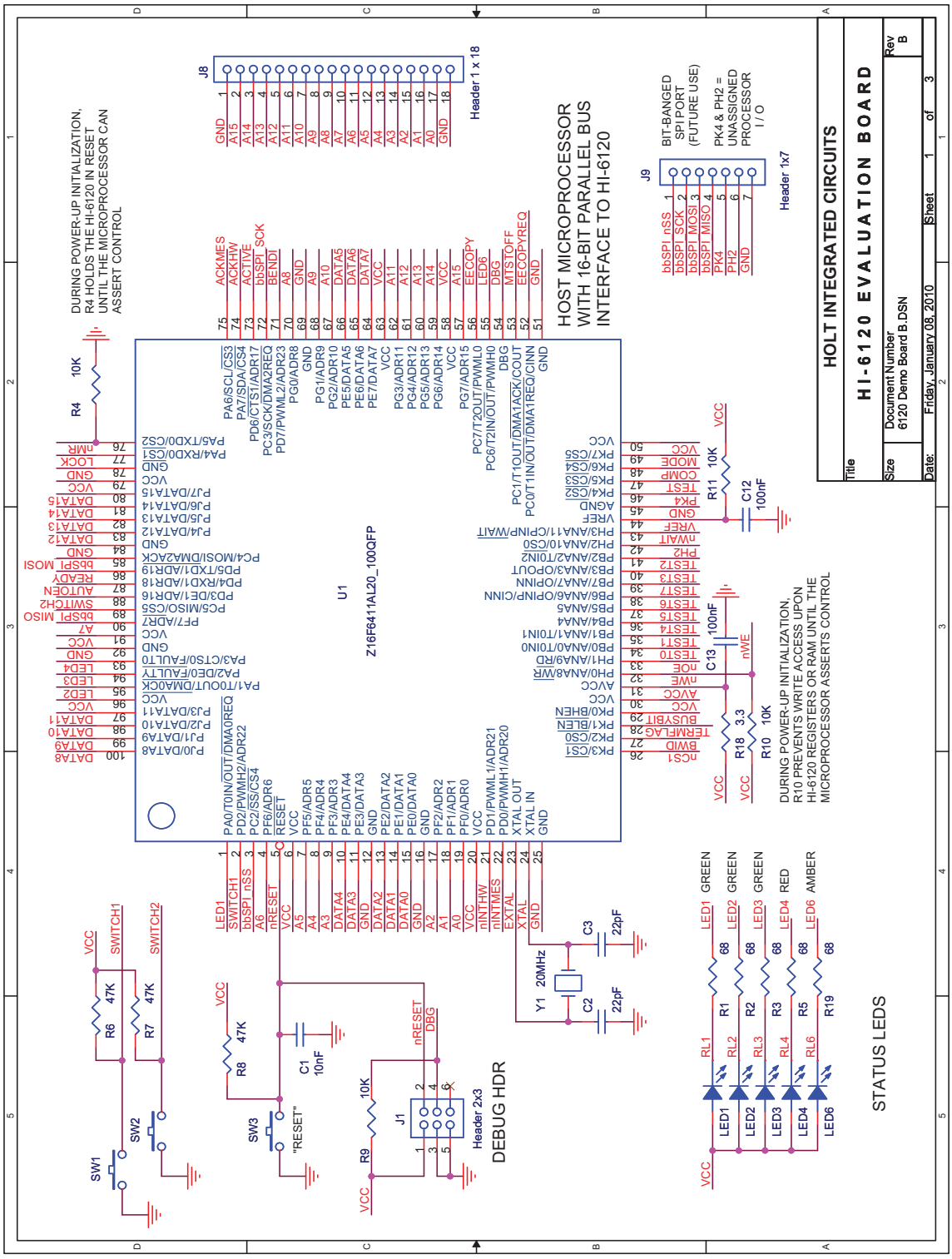


Figure 1. HI-6120 Schematic, Sheet 1 of 3.

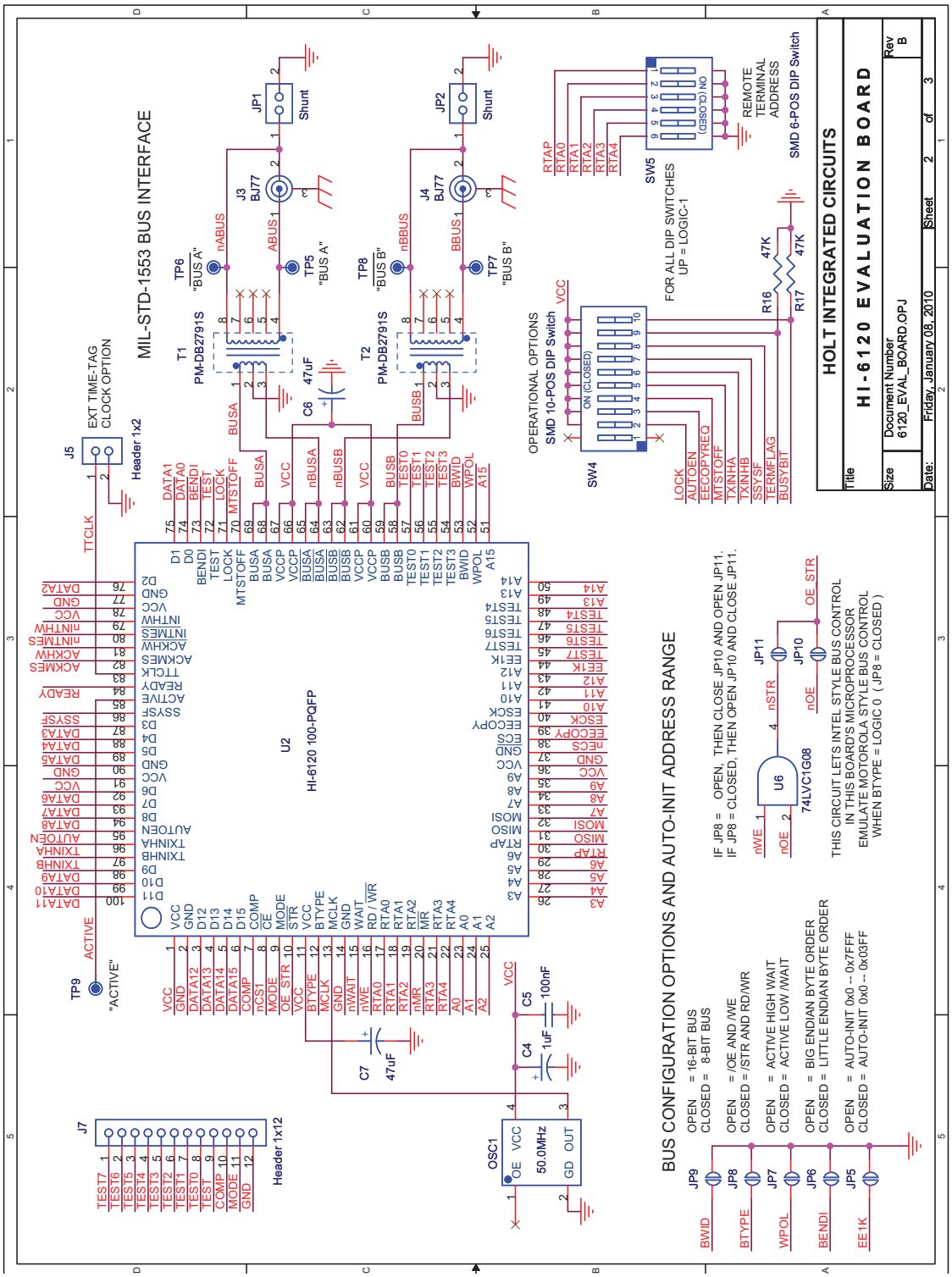


Figure 2. HI-6120 Schematic, Sheet 2 of 3.

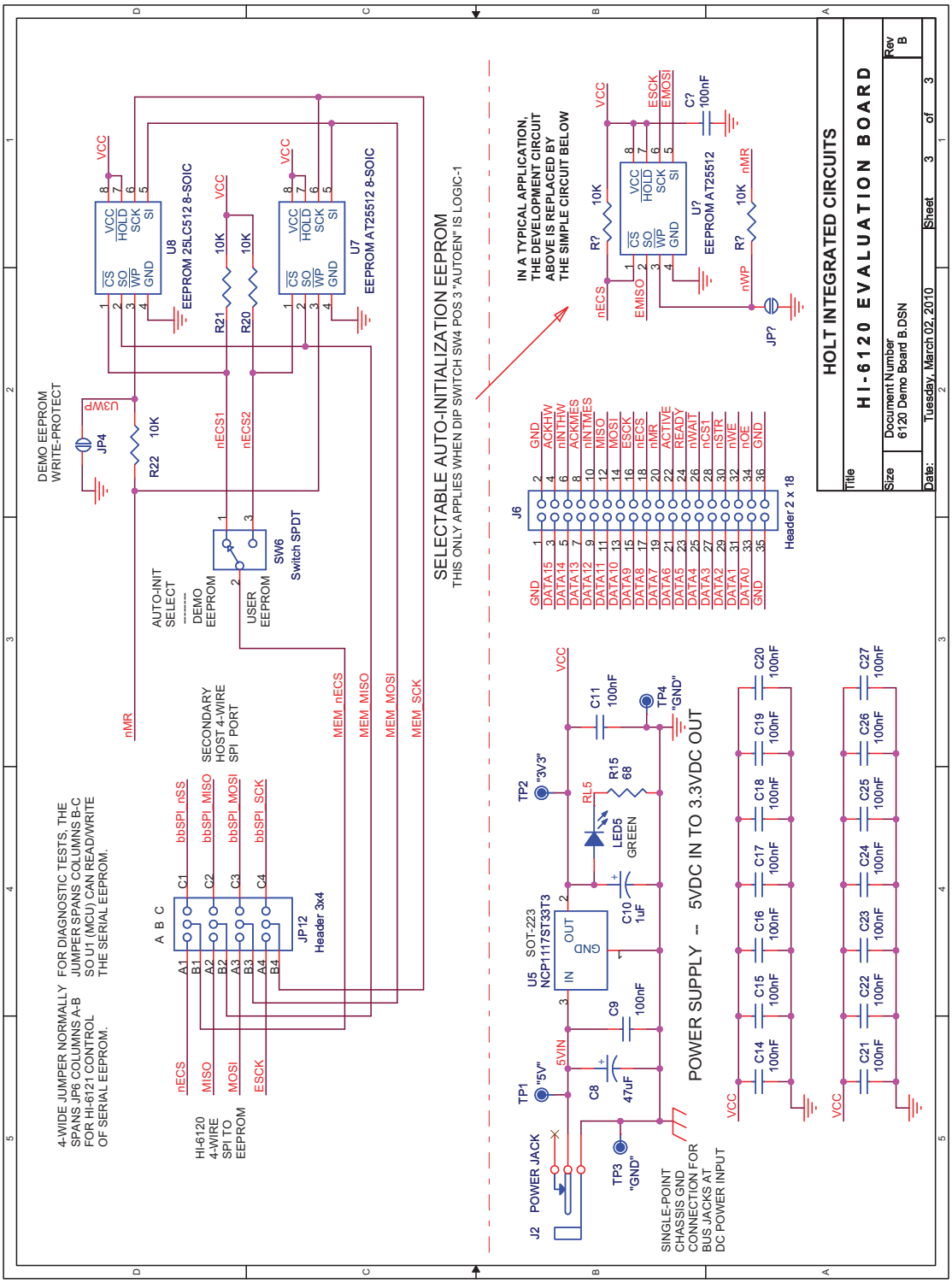


Figure 3. HI-6120 Schematic, Sheet 3 of 3.

10. BILL OF MATERIALS: HI-6121 DEVELOPMENT KIT

Item	*Qty*	*Description*	*Reference*	*DigiKey*	*MFR P/N*
1	1	PCB, Bare, Evaluation Board	N/A	--------	
2	1	Capacitor, Ceramic 10nF 10% 50V X7R 0805	C1	399-1158-1-ND	Kemet C0805C103K5RACTU
3	2	Capacitor, Ceramic 22pF 5% NPO 50V 0805	C2,C3	311-1103-1-ND	Yaego 0805CG220J9B200
4	2	Capacitor, Ceramic 1uF 10% 16V X7R 0805	C4,C10	399-1284-1-ND	Kemet C0805C105K4RACTU
5	15	Capacitor, Ceramic 0.1uF 20% 50V Z5U 0805	C5, C9, C11 - C21, C28,C29	399-1176-1-ND	Kemet C0805C104M5UACTU
6	3	Capacitor 68uF 10% 6.3V Tantalum SMD EIA 6032-28	C6,C7,C8	495-1507-1-ND	Kemet B45197A1686K309
7	1	Dual Schottky Diode 70V 70mA Common Cathode, SOT-23	D1	BAS7005INCT-ND	Infineon BAS 70-05 E6327
8	0	Header, Male 1x18, 0.1" Pitch	J6	S1012-18-ND DO NOT STUFF	Sullins
9	0	Header, Male 1x12, 0.1" Pitch	J7	S1012-12-ND DO NOT STUFF	Sullins
10	1	Header, Male 1x7, 0.1" Pitch	J8	S1012-7-ND	Sullins
11	1	Shunt, 0.1" with Handle	J8	A26853-ND	Tyco 4-881545-2
12	1	Receptacle, 0.1" Pitch, Right Angle 2x3	J1	S5518-ND	Sullins PPTC042LJBN-RC
13	1	Header, Male 2x3, 0.1" Pitch, 0.310" pins one side, 0.320" pins other side	J1	Programming Adapter for J1 to Zilog® Debug Cable	Samtec TSW-103-16-T-D
14	1	Jack, DC Power, 2.5mm ID x 2.1mm pin	J2	CP-102AH-ND	CUI PJ-102AH
15	2	Connector 3-Lug Concentric Triax Bayonet Jack, Panel Front Mount TRB	J3, J4	MilesTek 10-06570	Trompeter Electronics BJ77 Use 0.469 Round Hole
16	0	Header, Male, 0.1" Pitch, 1x2	J5	DO NOT STUFF	
17	0	Header, Male 1x2 0.1" Pitch, 0.230" Pins, 0.120" Tails	JP1,JP2	DO NOT STUFF	Sullins
18	0	Solder Jumper	JP5,JP8,JP9	DO NOT SOLDER	
19	1	Header, Male, 0.1" Pitch, 3x4	JP6	Samtec	Samtec TSW-104-07-T-T
	1	Shunt, 4-parallel, 0.1" pitch	JP6	Samtec	Samtec MNT-104-BK-T
20	4	LED Green 0805	LED1 - LED3, LED5	160-1179-2-ND	LiteOn LTST-C170GKT
21	1	LED Yellow 0805	LED6	160-1175-2-ND	LiteOn LTST-C170YKT
22	1	LED Red 0805	LED4	160-1176-2-ND	LiteOn LTST-C170CKT

Item	*Qty*	*Description*	*Reference*	*DigiKey*	*MFR P/N*
23	1	Oscillator, SMD 5 x 7 mm Program for 50.0MHz, 3.3V	OSC1	CSX-750PBBCT-ND	Citizen CSX-750P1P-UT, Program for 50.0 MHz
24	1	Resistor, 3.3 Ohm 5% 1/8W 0805	R10	311-3.3ARCT-ND	Any
25	6	Resistor, 68 5% 1/8W 0805	R1,R2,R3,R5, R15, R18	311-68ARCT-ND	Any
26	6	Resistor, 10K 5% 1/8W 0805	R4,R9,R11, R12,R13,R14	311-10KARCT-ND	Any
27	5	Resistor, 47K 5% 1/8W 0805	R6,R7,R8, R16,R17	311-47KARCT-ND	Any
28	3	Pushbutton	SW1,SW2,SW3	P10886SCT-ND	Panasonic EVQ-QWS02W
29	1	DIP Switch 10-Pos Slide SMD	SW4	CT1910LPST-ND	CTS 219-10LPST
30	1	DIP Switch 6-Pos Slide SMD	SW5	CT2196LPST-ND	CTS 219-6LPST
31	1	Slide Switch SPDT SMD	SW6	563-1022-1-ND	Copal CJS-1200TB
32	2	Transformer MIL-STD-1553 Single Bus 1:2.50 ratio	T1,T2	Holt PM-DB2791S	
33	2	Test Point, Red Insulator, 0.062" hole	TP5,TP7	5010K-ND	Keystone 5010
34	3	Test Point, Black Insulator, 0.062" hole	TP3,TP6,TP8	5011K-KD	Keystone 5011
35	1	Test Point, Orange Insulator, 0.062" hole	TP1	5008K-ND	Keystone 5008
36	1	Test Point, Yellow Insulator, 0.062" hole	TP2	5009K-ND	Keystone 5009
37	1	Test Point, White Insulator, 0.062" hole	TP9	5012K-ND	Keystone 5012
38	1	Test Point, Compact SMT	TP4	5016KCT-ND	Keystone 5016
39	1	IC, Z16F2810AG20SG 16-Bit MCU, 128K Flash 0-70C 64-LQFP	U1	269-4535-ND	Zilog® Z16F2810AG20SG
40	1	IC, HI-6121 52-PQFP	U2	Holt HI-6121PQI	
41	2	IC, Serial EEPROM 512Kbit 20MHz SPI 8-SOIC, Microchip	U3, U4	25LC512T-I/SNCT-ND	Microchip 25LC512T-I/SN
42	1	IC Voltage Regulator 3.3V 1A LDO, SOT-223	U5	497-1228-1-ND	ST Micro LD1117AS33TR
43	2	Pin Socket .015 to .021" pin, Use 0.040" hole	Y1	ED5037-ND	MillMax 0552-1-15-01-1127100
44	1	Crystal 20.00MHz, 50ppm 20pF, HC-49US leaded	Y1	631-1111-ND	Fox FOXSLF/S200-20
45	4	Stand-off, #4-40 Female Thread, 1" long	-----	Any	
46	4	Machine Screw, #4-40 x 1/4"	-----	Any	
47	4	Lock Washer, Int.Tooth #4-40	-----	Any	

11. SCHEMATICS: HI-6121 DEVELOPMENT KIT

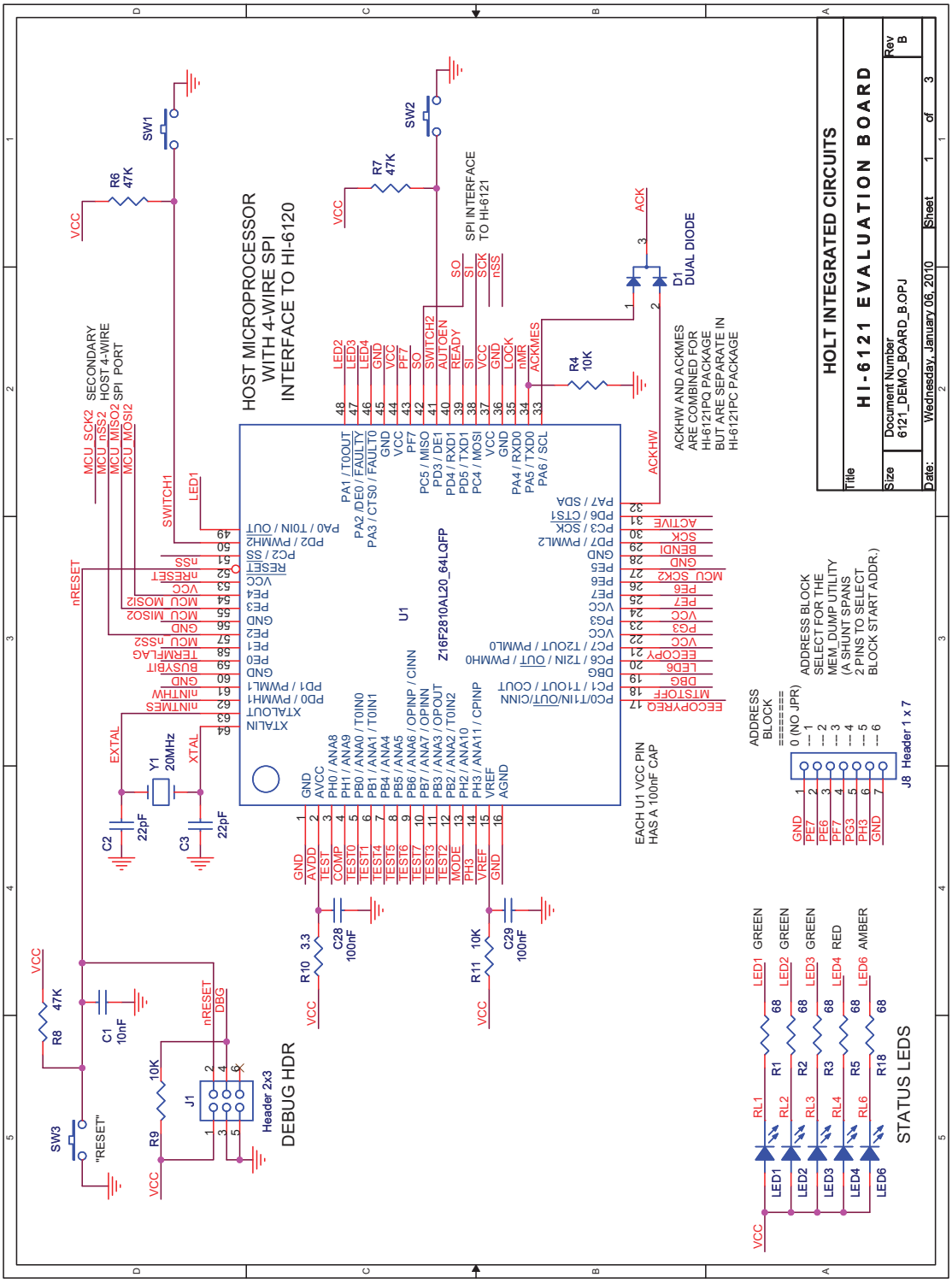


Figure 4. HI-6121 Schematic, Sheet 1 of 3.

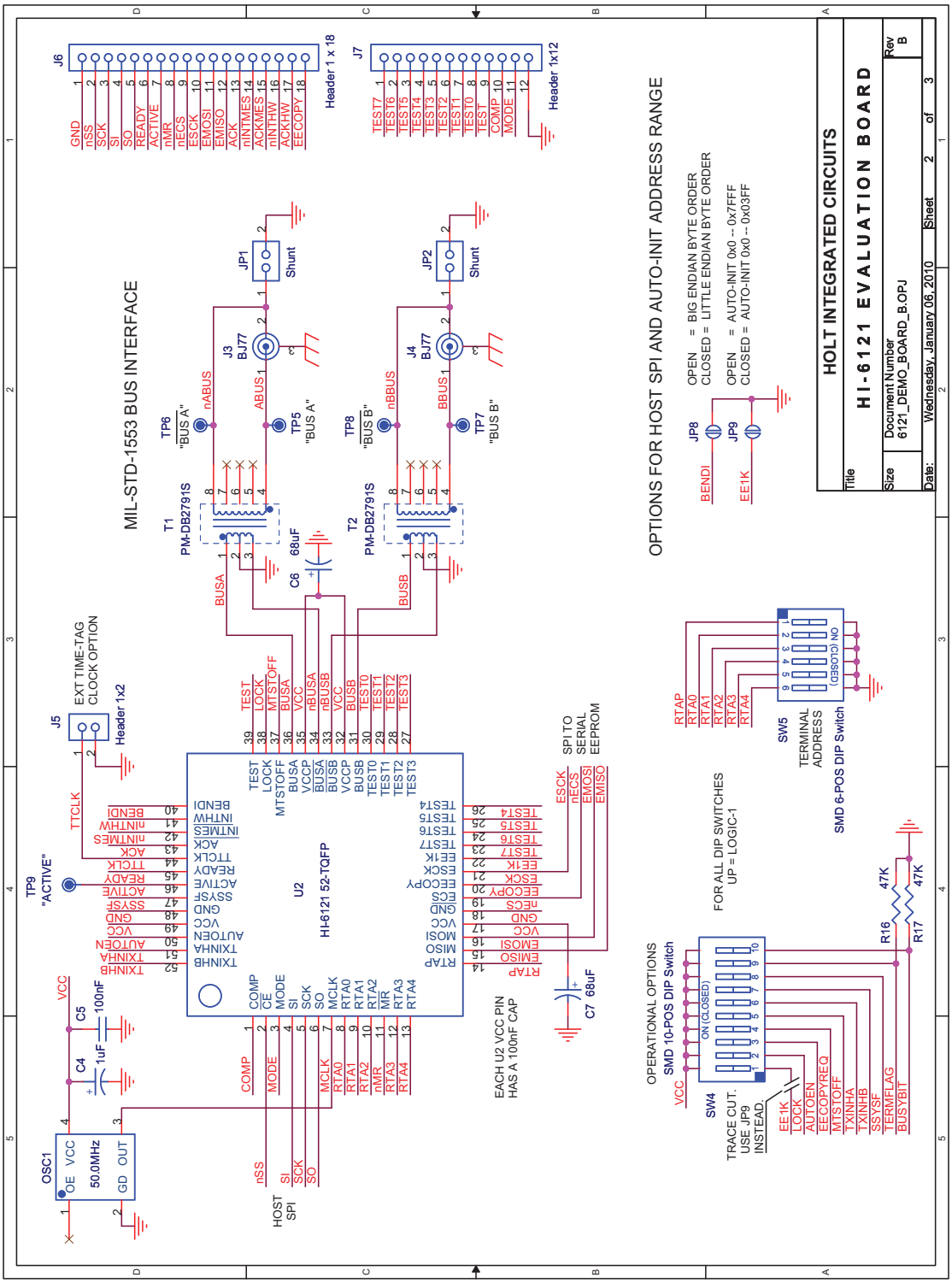


Figure 5. HI-6121 Schematic, Sheet 2 of 3.

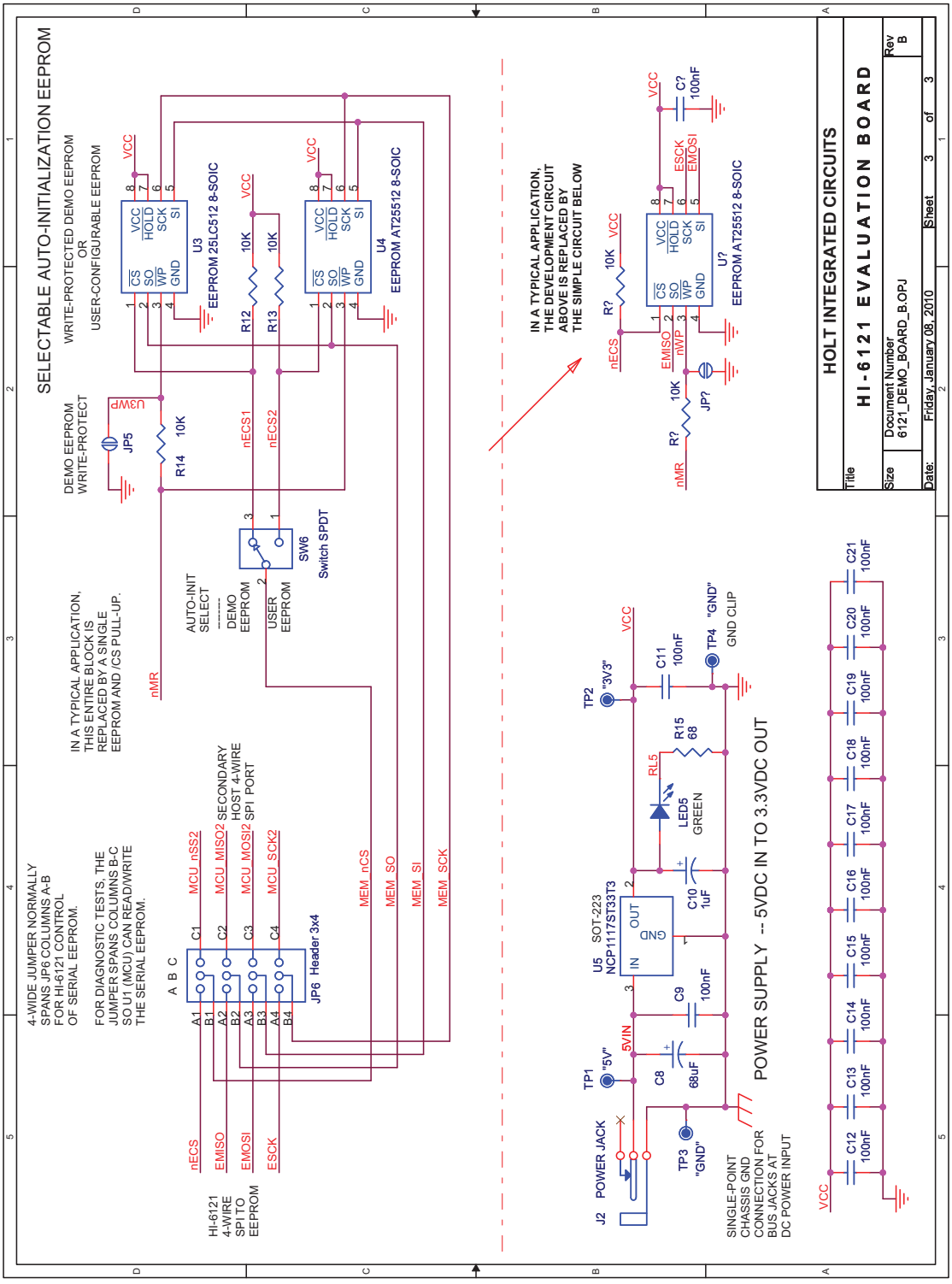
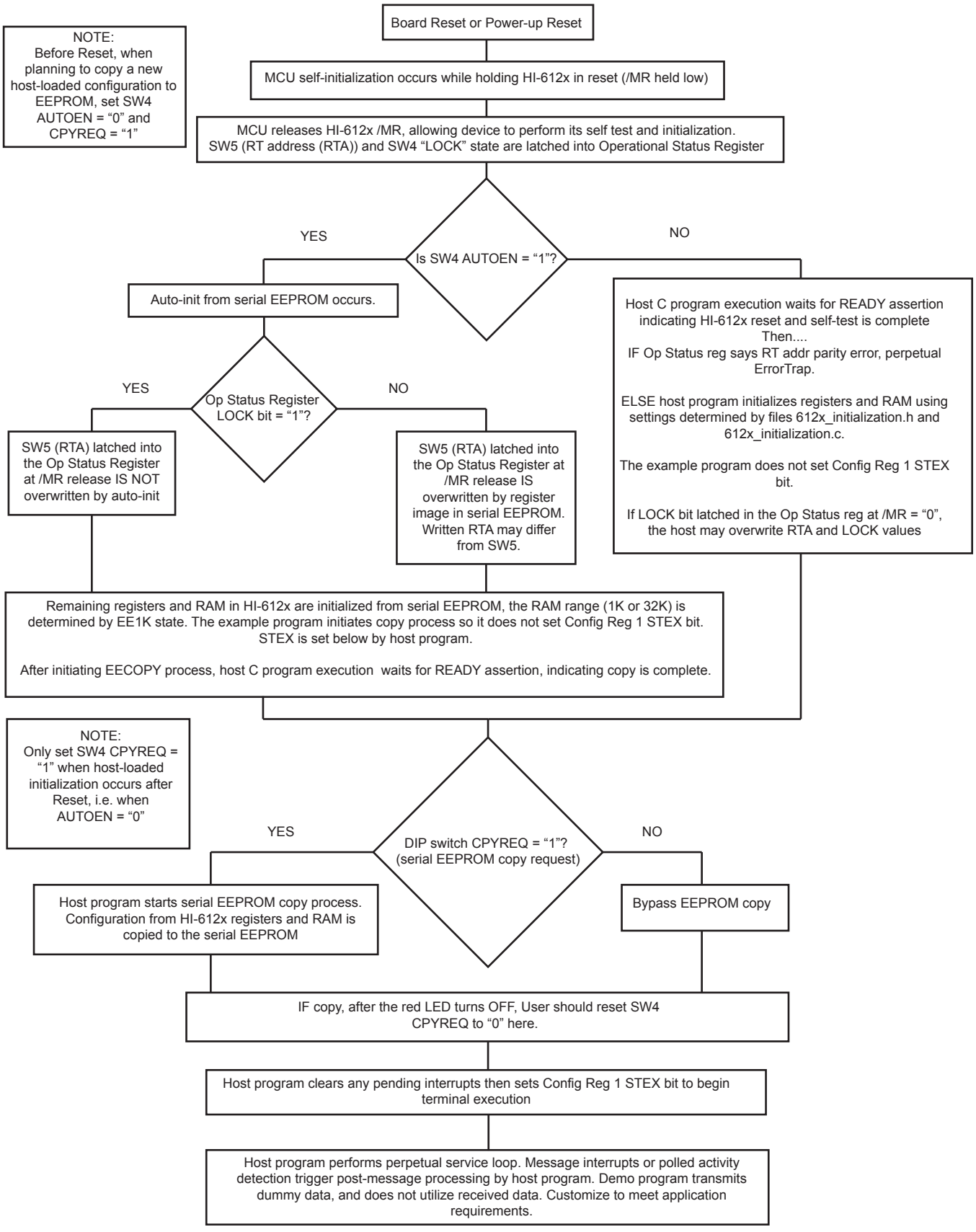


Figure 6. HI-6121 Schematic, Sheet 3 of 3.

12. INITIALIZATION SEQUENCE



13. REVISION HISTORY

Revision	Date	Description of Change
AN-6120, Rev. NEW	11/24/09	Initial Release.
Rev. A	3/5/10	Modified schematic. Updated BOM. Eliminated Errata from previous version and updated software to Rev 2.2.