

### INTRODUCTION

The Holt HI-8582 is a single-chip ARINC 429 interface IC incorporating two ARINC 429 receive channels and a single transmitter. Analog line drivers and receivers are included to allow the user to connect directly to the bus. The HI-8582 interfaces to a host CPU via a 16-bit bi-directional data bus. Discrete write and read lines are used to control the flow of information between the host and the HI-8582. Automatic label recognition, transmit and receiver protocol options and 32-word data buffering FIFOs are included on-chip. A complete description of the HI-8582 family can be found at the Holt web-site [www.holtic.com](http://www.holtic.com).

This applications note describes a simple interface design between the HI-8582 and an NXP (formerly Philips) XA-G49 microcontroller, a 16-bit derivative of the industry-standard Intel 8051 architecture. Assembly language code routines are shown as examples of how a user may program the configuration of the HI-8582, pre-load and verify ARINC 429 labels for auto-recognition, transmit data, poll status and retrieve received ARINC 429 data from the receivers.

In this example, the HI-8583PQI-10 version of the product is used.

### HARDWARE DESIGN

The example circuit is shown in figure 1. An RS-232 interface is used to program the XA-G49 internal EEPROM program memory. A simple LED port is also shown, and used in some of the code sequences as a pass/fail indicator.

The HI-8583 databus and control lines are wired to the XA-G49 parallel ports as shown in the following table. This chart may be used to interpret the assembly language code.

16-bit Microcontroller	HI-8583PQI-10
P2.7 - P2.0	DB15 - DB08
P0.7 - P0.0	DB07 - DB00
P1.7	$\overline{PL1}$
P1.6	$\overline{CWSTR}$
P1.5	$\overline{EN2}$
P1.4	$\overline{EN1}$
P1.3	SEL
P1.2	$\overline{DR2}$
P1.1	$\overline{DR1}$
P1.0	TX/R
P3.4	ENTX
P3.3	RSR
P3.2	$\overline{PL2}$

Note: - The example code provided only operates with receiver 1.  
 - The HI-8583PQI-10  $\overline{MR}$  is not controlled by software. A hardware reset is needed before use.

### BASIC OPERATION

#### Control Word

The control word is a 16-bit register used to configure the device. The control word register bits, CR15-CR0 are loaded from BD15-BD00 when  $\overline{CWSTR}$  is pulsed low. In the example program (8583-1word.axa), CR4 = 1, CR5 = 1 and CR15 = 1, enabling the 32<sup>nd</sup> transmitter bit as parity, running in normal operation and unscrambling the ARINC data.

The following code loads the control word.

```
;Set control word
mov     P2, #80H
mov     P0, #30H
clr     p1.6; cwstr*
setb    p1.6
```

#### Transmitter FIFO

To load the transmitter FIFO,  $\overline{PL1}$  and  $\overline{PL2}$  are used for the first and second bytes respectively. To load byte one on BD15-BD00,  $\overline{PL1}$  is pulsed low, followed by byte two on BD15-BD00 and  $\overline{PL2}$  pulsed low.

The following code loads one word in the transmitter FIFO:

```
Txfifo:
;Loading transmitter FIFO
;Word 1
mov     P2, #00H
mov     P0, #00H
clr     P1.7 ;PL1*
setb    P1.7
mov     P2, #01H
mov     P0, #01H
clr     P3.2 ;PL2*
setb    P3.2
```

#### Transmitting TX FIFO

When the TX FIFO contains at least one word, the TX/R signal goes low. In this example, data transmission begins after one word is loaded. To transmit the contents of the TX FIFO, ENTX is set high.

The following code verifies TX/R is low before ENTX is set high. If TX/R is high, the program will re-load the TX FIFO. When TX/R has gone from low to high the program lowers ENTX.

```
;Transmitter FIFO contains 1 word
jnb     P1.0, transmit
;if TX/R is low, then go to transmit if
;not then
jmp     txfifo

Transmit:
Setb    P3.4
;enable ENTX to transmit TX FIFO
jnb     P1.0, transmit
;if TX/R is low, then go to transmit,
;if not then
clr     P3.4
```

## BASIC OPERATION (cont.)

### Unload RX FIFO

When the RX FIFO contains at least one valid ARINC word, the corresponding  $\overline{D/R}$  signal will go low. To retrieve the data in receiver 1, SEL is low for byte 1 and  $\overline{EN1}$  is pulsed low. To retrieve byte 2, SEL is high and  $\overline{EN1}$  is pulsed low.

In this example the retrieved word is compared to the transmitted word to confirm there were no errors in transmitting and receiving. A pass or fail LED indicates the outcome of the comparison.

The following code verifies the status of  $\overline{D/R1}$ , unloads one ARINC word and verifies it against the transmitted word.

```

rxtest:
  mov     R0L, #00H
  jnb    P1.1, unRX1
  ;if D/R1* is low, then unload RX 1
  ;if not ready to unload, then wait
  jmp    rxtest

readPorts:
  CJNE   R0L, P2, fail
  CJNE   R0L, P0, fail
  ret

unRX1:
  mov.b  P0CFGA, #00H ;push pull
  mov.b  P2CFGA, #00H ;push pull

  clr    P1.3 ;lower SEL for first
  clr    P1.4 ;activating EN1*
  call   readPorts
  setb   P1.4
  mov    R0L, #01H
  setb   P1.3 ;2nd byte SEL high
  clr    P1.4 ;activating EN1*
  call   readPorts
  setb   P1.4 ;deactivate EN1*

pass:
  setb   P3.7 ;pass LED
  jmp    finished

fail:
  setb   P3.5 ;fail LED
  jmp    finished

finished:
  end

```

## LABELS

To use label recognition all 16 labels must be pre-loaded in label memory. When writing to, or reading from label memory, SEL must be high, all 16 locations should be accessed and CR1 must be written a zero before returning to normal operation. When reading labels from memory, label recognition must be disabled ( $CR2/3 = 0$ ).

### Setting Control Word

Before loading 16 labels into memory, CR1, must have been a zero prior to being written a one, to initiate the label memory sequence.

The following code ensures CR1 was a zero prior to a one.

```

;Set control word (CR1 from 0 to 1)
  mov    P2, #00H
  mov    P0, #00H
  clr    p1.6
  setb   P1.6
  mov    P2, #80H
  mov    P0, #02H
  clr    p1.6; cwstr*
  setb   P1.6

```

### Loading Labels

$\overline{PL1}$  and  $\overline{PL2}$  are used to load labels into memory associated with receivers 1 and 2, respectively. Label data is loaded into memory using databus pins BD07-BD00. To load labels, data must be present on BD07-BD00 and  $\overline{PL1}/\overline{PL2}$  pulsed low while SEL is high.

The following code loads a label in label memory associated with receiver 1.

```

Labels:
;loading label memory with one label
  mov    P0, #01H
  clr    P1.7 ;PL1*
  setb   P1.7

```

### Re-setting Control Word

After 16 labels have been loaded the control word must return CR1 to normal operation. In the example program (labels.axa), after loading 16 labels, label memory is read back. To read label memory, CR1 must be switched from a zero to a one.

The following code writes a zero to CR1 when finished loading labels then writes a one to CR1 to read labels.

```

;label memory contains 16 words
;Set CR1 to normal operation
  mov    P2, #80H
  mov    P0, #00H
  clr    p1.6 ;cwstr*
  setb   P1.6

;Set CR1 from 0 to 1 to read labels
  mov    P2, #80H
  mov    P0, #02H
  clr    p1.6; cwstr*
  setb   P1.6

```

### Reading Labels

To read labels,  $\overline{EN1}$  and  $\overline{EN2}$  are used for receivers 1 and 2 respectively. Label data is output on BD07-BD00. After CR1 has gone from zero to one, while SEL is high, a low pulse on  $\overline{EN1}$  or  $\overline{EN2}$  will output the first label on BD07-BD00. All 16 labels must be read before writing CR1 to a zero.

The following code reads the first label from memory associated with receiver 1.

```

unlabels:
  mov.b  P0CFGA, #00H ;push pull
  mov    R0L, #01H
  clr    P1.4 ;activating EN1*
  CJNE   R0L, P0, fail

```

## STATUS REGISTER

The HI-8582 and HI-8583 contain a 9-bit status register which can be polled to determine the status of the receiver FIFOs and transmitter FIFO. To poll the status register, SEL is low, and  $\overline{RSR}$  is pulsed low. The status register contents are output on BD08-BD00. The unused bits are output as zeroes.

The following code is an example of how to unload the contents of the status register and compare them to a known value.

```

mov     R0L, #40H
mov     R0H, #00H
clr     P1.3 ;SEL low
clr     P3.3 ;RSR* low
nop     ;stabilize data before reading
nop
CJNE   R0L, P0, failfar
CJNE   R0H, P2, failfar
setb   P3.3 ;RSR* back to high

```

## Additional Information

Information on the NXP 16-bit XA-G49 microprocessor can be found by searching the device type number (or 16-bit XA microcontroller family) at [www.nxp.com](http://www.nxp.com).

The software assembler used in this design example is the "Software Development Suite for XA" offered by Raisonance S.A.S.. A free evaluation version "EvalXA" has an 8KB maximum program size. More information can be found at [www.raisonance.com](http://www.raisonance.com).

In-system flash programming was accomplished using the free Flash Magic utility from Embedded Systems Academy. This utility can be downloaded at [www.flashmagictool.com](http://www.flashmagictool.com).

## EXAMPLE PROGRAMS

The '8583-1word.axa' example program configures the HI-8583-10, loads one word into the TX FIFO, transmits the word, unloads the RX FIFO and compares the received word to the original transmitted word.

The HI-8583PQI-10 is configured to enable the 32nd transmitter bit as parity, run in normal operation and unscramble the ARINC data. The microcontroller is programmed to verify status flags before proceeding to transmit and unload the RX FIFO.

### 8583-1word.axa

\$pagewidth (132t)  
\$Listing\_min

```
=====
;
;           8583-1word.AXA
;
=====
```

```
$include (xa-g3.inc)
$include (regxag49.inc)
```

```
org 0
dw 8f00H,start
org 120H
```

```
start:
clr    P3.5    ;fail LED
clr    p3.7    ;pass LED
clr    P1.3    ;SEL
clr    P3.4    ;ENTX
```

```
;initialize XA-G49 port pins
;data bus pins
mov.b  P0CFGB, #0FFH ;push pull
mov.b  P0CFGA, #0FFH ;push pull
mov.b  P2CFGB, #0FFH ;push pull
mov.b  P2CFGA, #0FFH ;push pull
;various input and output pins
mov.b  P3CFGB, #40H
mov.b  P3CFGA, #0BFH
mov.b  P1CFGB, #07H
mov.b  P1CFGA, #0F8H
```

```
program:
;Initialize 8582 input pins
setb   P1.7    ;set PL1*
setb   P1.6    ;set CWSTR*
setb   P1.5    ;set EN2*
setb   P1.4    ;set EN1*
setb   P3.3    ;set RSR*
setb   P3.2    ;set PL2*
```

```
;Set control word
mov.b  P2, #80H
mov    P0, #30H
clr    p1.6; cwstr*
setb   p1.6
```

```
txfifo:
;Loading transmitter FIFO
;Word 1
mov    P2, #00H
mov    P0, #00H
clr    P1.7 ;PL1*
setb   P1.7
mov    P2, #01H
mov    P0, #01H
```

```
clr    P3.2 ;PL2*
setb   P3.2
nop
nop
```

```
;TX FIFO contains 1 word
jnb    P1.0, transmit ;if TX/R is low, then
;go to transmit if not then
jmp    txfifo

transmit:
setb   P3.4 ;enable ENTX to transmit TX FIFO
jnb    P1.0, transmit ;if TX/R is low, then go
;to transmit if not then
clr    P3.4
```

```
rxtest:
mov    R0L, #00H
jnb    P1.1, unRX1 ;if D/R1* is low, then
;unload RX 1
;if neither are ready to unload, then
;wait
jmp    rxtest
```

```
readPorts:
CJNE   R0L, P2, fail
CJNE   R0L, P0, fail
ret
```

```
unRX1:
mov.b  P0CFGA, #00H ;push pull
mov.b  P2CFGA, #00H ;push pull
clr    P1.3 ;lower SEL for first byte
clr    P1.4 ;activating EN1*
call   readPorts
setb   P1.4
mov    R0L, #01H
setb   P1.3 ;2nd byte SEL high
clr    P1.4 ;activating EN1*
call   readPorts
setb   P1.4 ;deactivate EN1*
```

```
pass:
setb   P3.7 ;pass LED
jmp    finished
```

```
fail:
setb   P3.5 ;fail LED
jmp    finished
```

```
finished:
end
```

## EXAMPLE PROGRAMS (cont.)

The 'labels.axa' example program writes 16 labels in to memory and reads the labels back while verifying each one to the original.

Before writing 16 labels CR1 is written from '0' to '1' to initiate the label memory sequence. After writing 16 labels, CR1 is written a '0' for normal operation. Before reading the labels, CR1 is written from '0' to '1' to initiate the label memory sequence. After all labels are read and compared to the originals, the pass/fail LEDs will indicate the outcome.

### Labels.axa

```
$pagewidth (132t)
```

```
$Listing_min
```

```
=====
;                               labels.AXA
;                               =====
```

```
$include (xa-g3.inc)
```

```
$include (regxag49.inc)
```

```
org 0
dw 8f00H,start
org 120H
```

```
start:
```

```
clr    P3.5    ;fail LED
clr    p3.7    ;pass LED
clr    P1.3    ;SEL
clr    P3.4    ;ENTX
```

```
;initialize XA-G49 port pins
```

```
;data bus pins
```

```
mov.b  P0CFGB, #0FFH ;push pull
```

```
mov.b  P0CFGA, #0FFH ;push pull
```

```
mov.b  P2CFGB, #0FFH ;push pull
```

```
mov.b  P2CFGA, #0FFH ;push pull
```

```
;various input and output pins
```

```
mov.b  P3CFGB, #40H
```

```
mov.b  P3CFGA, #0BFH
```

```
mov.b  P1CFGB, #07H
```

```
mov.b  P1CFGA, #0F8H
```

```
program:
```

```
;Initialize 8582 input pins
```

```
setb   P1.7    ;set PL1*
setb   P1.6    ;set CWSTR*
setb   P1.5    ;set EN2*
setb   P1.4    ;set EN1*
setb   P3.3    ;set RSR*
setb   P3.2    ;set PL2*
setb   P1.3    ;set SEL for label writing
```

```
;Set control word (CR1 from 0 to 1)
```

```
mov    P2, #00H
mov    P0, #00H
clr    p1.6
setb   P1.6
mov    P2, #80H
mov    P0, #02H
clr    p1.6 ;cwstr*
setb   p1.6
```

```
labels:
```

```
;loading label memory with 16 labels
```

```
mov    P0, #01H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #02H
```

```
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #03H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #04H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #05H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #06H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #07H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #08H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #18H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #28H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #38H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #48H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #58H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #68H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #78H
clr    P1.7 ;PL1*
setb   P1.7
mov    P0, #88H
clr    P1.7 ;PL1*
setb   P1.7
```

```
;label memory contains 16 words
```

```
;Set CR1 to normal operation
```

```
mov    P2, #80H
mov    P0, #00H
clr    p1.6
setb   P1.6
```

```
;Set CR1 to 0 then to 1 to read labels
```

```
mov    P2, #80H
mov    P0, #02H
clr    p1.6 ;cwstr*
setb   p1.6
```

```
unlabels:
```

```
mov.b  P0CFGA, #00H ;push pull
mov    R0L, #01H
clr    P1.4; activating EN1*
CJNE   R0L, P0, fail
setb   P1.4;
mov    R0L, #02H
clr    P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov    R0L, #03H
clr    P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov    R0L, #04H
clr    P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
```

## EXAMPLE PROGRAMS (cont.)

### Labels.axa (cont.)

```

mov     R0L, #05H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #06H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #07H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #08H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #18H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #28H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #38H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #48H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #58H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #68H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #78H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*
mov     R0L, #88H
clr     P1.4;activating EN1*
CJNE   R0L, P0, fail
setb   P1.4; deactivate EN1*

pass:
setb   P3.7 ;pass LED
jmp    finished

fail:
setb   P3.5 ;fail LED
jmp    finished

finished:
end

```

## EXAMPLE PROGRAMS (cont.)

The 'FIFOSR.axa' example program demonstrates the use of the status register while loading 32 words in the transmitter FIFO and unloading the 32 transmitted words from the receiver FIFO.

'FIFOSR.axa' sets the control word to enable the 32nd transmitter bit to parity, operate in normal operation and unscramble the ARINC data. The status register is read before any tasks are performed to verify all FIFOs are empty. The status register is read after 15 words are loaded in the TX FIFO. The 16th word is loaded, and the status register is read before another 16 words are loaded to fill the FIFO with 32 words. All 32 words are transmitted and the status register is read to verify the TX FIFO is empty, RX1 is full and RX2 is empty. The status register is read after 1 word is unloaded, to verify the RX1 FIFO is no longer full. The status register is read after another 16 words are unloaded. Every unloaded word is compared to the transmitted words to check for errors. The remaining words are unloaded and the pass/fail LED indicates the outcome of the comparisons.

### FIFOSR.axa

```
$pagewidth (132t)
$Listing_min
```

```
;=====
;                      FIFOSR.AXA
;=====
```

```
$include (xa-g3.inc)
$include (regxag49.inc)
```

```
org 0
dw 8f00H,start
org 120H
```

```
start:
```

```
mov.b wdcon, #0H
mov.b wfeed1, #0A5H
mov.b wfeed2, #5AH
```

```
beg:
```

```
clr P3.5 ;fail LED
clr P3.6 ;other fail
clr P3.7 ;pass LED
clr P1.3 ;SEL
clr P3.4 ;ENTX
```

```
;initialize XA-G49 port pins
```

```
;data bus pins
```

```
mov.b P0CFGB, #0FFH ;push pull
mov.b P0CFGA, #0FFH ;push pull
mov.b P2CFGB, #0FFH ;push pull
mov.b P2CFGA, #0FFH ;push pull
```

```
;various input and output pins
```

```
mov.b P3CFGB, #00H
mov.b P3CFGA, #0FFH
mov.b P1CFGB, #07H
mov.b P1CFGA, #0F8H
```

```
program:
```

```
;Initialize 8582 input pins
```

```
setb P1.7 ;set PL1*
setb P1.6 ;set CWSTR*
setb P1.5 ;set EN2*
setb P1.4 ;set EN1*
setb P3.3 ;set RSR*
setb P3.2 ;set PL2*
```

```
;Set control word
```

```
mov P2, #80H
mov P0, #30H
clr p1.6; cwstr*
setb p1.6
```

```
;read status register
```

```
mov.b P0CFGA, #00H ;input
mov.b P2CFGA, #00H ;input
mov R0L, #40H
mov R0H, #00H
clr P1.3 ;SEL high
clr P3.3 ;RSR* low
nop ;stabilize data before reading
nop
CJNE R0L, P0, failfar
CJNE R0H, P2, failfar
setb P3.3 ;RSR* back to high
mov.b P0CFGA, #0FFH ;push pull
mov.b P2CFGA, #0FFH ;push pull
jmp txfifo
```

```
failfar:
```

```
jmp fail
```

```
txfifo:
```

```
;Loading transmitter FIFO
```

```
mov R0L, #1EH
mov R0H, #1EH
mov P2, #00H
mov P0, #00H
```

```
top:
```

```
clr P1.7 ;PL1*
setb P1.7
add P2, #01H
add P0, #01H
clr P3.2 ;PL2*
setb P3.2
```

```
add P2, #01H
add P0, #01H
CJNE R0L, P2, top
CJNE R0H, P0, top
```

```
;read status register at 15 words loaded
```

```
mov.b P0CFGA, #00H ;input
mov.b P2CFGA, #00H ;input
mov R0L, #00H
clr P1.3 ;SEL low
clr P3.3 ;RSR* low
CJNE R0L, P0, fail
CJNE R0H, P2, fail
setb P3.3 ;RSR* back to high
mov.b P0CFGA, #0FFH ;push pull
mov.b P2CFGA, #0FFH ;push pull
```

```
mov P2, #1EH
mov P0, #1EH
clr P1.7 ;PL1*
setb P1.7
mov P2, #1FH
mov P0, #1FH
clr P3.2 ;PL2*
setb P3.2
```

```
;read status register at 16 words loaded
```

```
mov.b P0CFGA, #00H ;input
mov.b P2CFGA, #00H ;input
mov R0L, #00H
mov R0H, #01H
clr P1.3 ;SEL low
clr P3.3 ;RSR* low
CJNE R0L, P0, fail
CJNE R0H, P2, fail
setb P3.3 ;RSR* back to high
mov.b P0CFGA, #0FFH ;push pull
mov.b P2CFGA, #0FFH ;push pull
```

## EXAMPLE PROGRAMS (cont.)

## FIFOSR.axa (cont.)

```

mov     R0L, #40H
mov     R0H, #40H
mov     P2, #20H
mov     P0, #20H

next16:
clr     P1.7 ;PL1*
setb   P1.7
add    P2, #01H
add    P0, #01H
clr    P3.2 ;PL2*
setb   P3.2

add    P2, #01H
add    P0, #01H
CJNE  R0L, P2, next16
CJNE  R0H, P0, next16
nop
nop

;TX FIFO contains 32 words
jnb    P1.0, transmit ;if TX/R is low,
;then go to transmit if not then
jmp    txfifo

transmit:
setb   P3.4 ;enable ENTX to transmit TX FIFO
jnb    P1.0, transmit ;if TX/R is low, then go
;to transmit if not then
clr    P3.4
jmp    rxtest

rxtest:
jnb    P1.1, unRX1 ;if D/R1* is low, then
;unload RX 1
;if not ready to unload, then wait
jmp    rxtest

readPorts:
nop
nop
CJNE  R0H, P2, fail
CJNE  R0L, P0, fail
ret

fail:
setb   P3.5; fail LED
setb   P3.6; other fail
jmp    endfail

unRX1:
delay100: ;delay
mov    r0, #15H
delay101:
djnz  r0, delay101

mov.b  P0CFGA, #00H ;push pull
mov.b  P2CFGA, #00H ;push pull

;read status register
mov    R0L, #47H
clr    P1.3 ;SEL low
clr    P3.3 ;RSR* low
nop    ;stabilize data before reading
nop
CJNE  R0L, P0, fail
setb   P3.3 ;RSR* back to high
mov    R0H, #00H
mov    R0L, #00H
clr    P1.3; lower SEL for first byte
clr    P1.4; activating EN1*
call  readPorts
setb   P1.4;
add    R0H, #01H

add    R0L, #01H
setb   P1.3; 2nd byte SEL high
clr    P1.4; activating EN1*
call  readPorts
setb   P1.4 ;deactivate EN1*

add    R0H, #01H
add    R0L, #01H
CJNE  R0L, #40H, readlast15
CJNE  R0H, #40H, readlast15

readlast15:
clr    P1.3; lower SEL for first byte
clr    P1.4; activating EN1*
call  readPorts
setb   P1.4;
add    R0H, #01H
add    R0L, #01H
setb   P1.3 ;2nd byte SEL high
clr    P1.4 ;activating EN1*
call  readPorts
setb   P1.4 ;deactivate EN1*

add    R0H, #01H
add    R0L, #01H
CJNE  R0L, #40H, readlast15
CJNE  R0H, #40H, readlast15

pass:
setb   P3.7; pass LED
jmp    finished

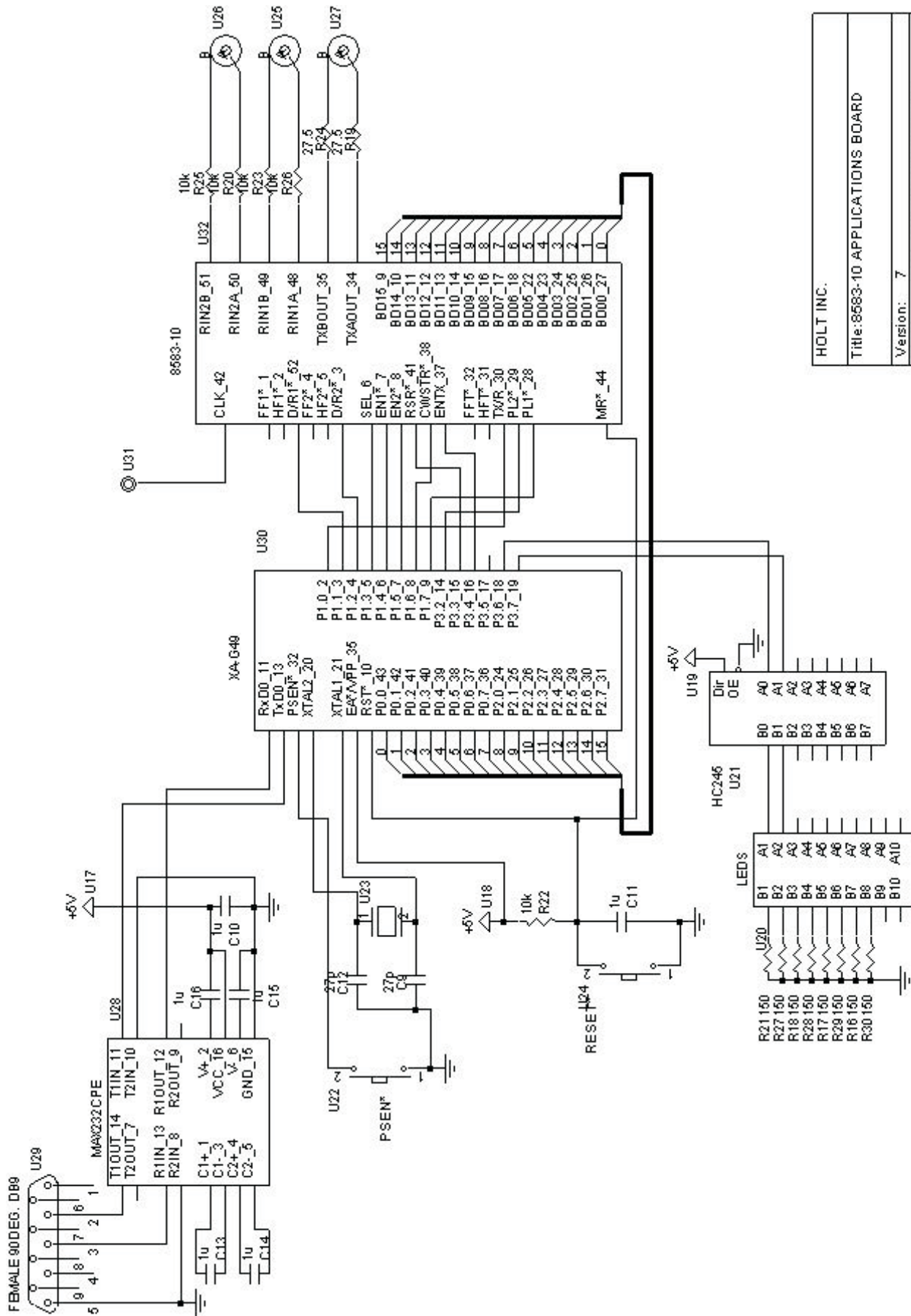
finished:
delay: ; delay
mov    r0, #640H
delay1:
djnz  r0, delay1

jmp    beg

endfail:
end

```

Figure 1: Example Circuit



HOLT INC.  
 Title: 8683-10 APPLICATIONS BOARD  
 Version: 7  
 Date: Friday, September 12, 2003  
 Drawn By: C. LAVALLEE