EXPANDED CONTROLLER INTERFACE DEVICE INPUT/OUTPUT CAPABILITIES AND CID SOFTWARE COORDINATION

Final Report
KLK235
N07-04

National Institute for Advanced Transportation Technology

University of Idaho

Ahmed Abdel-Rahim; Brian Johnson

Zhen Li; Eugene Bordenkircher; Joel Alberts; Sanjeev Giri; Mark Hutchinson

March 2007

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Expanded Controller Interface Device Input/Output Capabilities and CID Software Coordination		5. Report Date December 2006 6. Performing Organization Code KLK235
5.Author(s) Ahmed Abdel-Rahim; Brian Johnson with Zhen Li, Eugene Bordenkircher, Joel Alberts, Sanjeev Giri and Mark Hutchinson		8. Performing Organization Report No. N07-04
9. Performing Organization Name and Address National Institute for Advanced Transportation Technology University of Idaho PO Box 440901; 115 Engineering Physics Building Moscow, ID 838440901		10. Work Unit No. (TRAIS) 11. Contract or Grant No. *DTRS98-G-0027*
12. Sponsoring Agency Name and Address US Department of Transportation Research and Special Programs Administration 400 7th Street SW Washington, DC 20509-0001		13. Type of Report and Period Covered Final Report: August 2004-December 2005 14. Sponsoring Agency Code *USDOT/RSPA/DIR-1*
Supplementary Notes:		

16. Abstract

The controller interface device (CID) is the result of several years of hardware and software development by NIATT. This project had two objectives: the first was to expand the capability of the CID for applications where the number of input/output connections limit performance. The second objective was to investigate a new application area for CID technology, developing and testing a prototype to use the CID and CORSIM simulation to test traffic controller compliance to NTCIP communication standards.

This final report is made in two parts. The first describes the development of a synchronous data link control (SDLC) interface capability for the CID. The second part discusses the completion of the real-time playback system to test CID timing performance introduced in and is presented in the form of a paper titled: "Real-Time Playback Hardware-in-the-Loop Simulation of Traffic Systems," presented at IECON 2005, 32nd Annual Conference of the IEEE Industrial Electronics Society. This paper discusses the development of a software-controlled embedded system to evaluate the effect of communication latencies in hardware-in-the-loop simulation of traffic systems. The tool uses the Controller Interface Device (CID) hardware developed for hardware-in-the-loop simulation with modifications made to the firmware to support real-time playback (RTPB). RTPB simulators have been used in power systems as a cheap alternative to real-time simulators. In some cases RTPB is the only possible simulation method. This paper presents the application of RTPB to traffic simulations using actual traffic controller hardware.

17. Key Words	18. Distribution Statement
Hardware-in-the-loop simulation; traffic signal controllers; traffic simulation; real-time control	Unrestricted; Document is available to the public through the National Technical Information Service; Springfield, VT.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		…

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The controller interface device (CID) is the result of several years of hardware and software development by NIATT. This project had two objectives: the first was to expand the capability of the CID for applications where the number of input/output connections limit performance. The second objective was to investigate a new application area for CID technology, developing and testing a prototype to use the CID and CORSIM simulation to test traffic controller compliance to NTCIP communication standards.

This final report is made in two parts. The first describes the development of a synchronous data link control (SDLC) interface capability for the CID. The second part discusses the completion of the real-time playback system to test CID timing performance introduced in and is presented in the form of a paper titled: "Real-Time Playback Hardware-in-the-Loop Simulation of Traffic Systems," presented at IECON 2005, 32nd Annual Conference of the IEEE Industrial Electronics Society. This paper discusses the development of a software-controlled embedded system to evaluate the effect of communication latencies in hardware-in-the-loop simulation of traffic systems. The tool uses the Controller Interface Device (CID) hardware developed for hardware-in-the-loop simulation with modifications made to the firmware to support real-time playback (RTPB). RTPB simulators have been used in power systems as a cheap alternative to real-time simulators. In some cases RTPB is the only possible simulation method. This paper presents the application of RTPB to traffic simulations using actual traffic controller hardware.

PART 1: SYNCHRONOUS DATA LINK CONTROL INTERFACE FOR INTERFACE FOR CONTROLLER INTERFACE DEVICE TO TRAFFIC CONTROLLER COMMUNICATION

A. INTRODUCTION

The synchronous data link control (SDLC) controller interface device (CID) is an adaptation of the original CID. The SDLC version replaces the original microcontroller board with a modified board that incorporates the original board in addition to a connector that allows for SDLC communication to TS/2 standard traffic controllers (see Fig. 1a and 1b). Neither the motherboard nor the daughterboards need to be modified. The CID case will need an additional cut out for the series port.

The benefit in using the new connector is that the SDLC link replaces over 80 wires with a single nine-wire RJ-45 cable by using high speed serial communications. Also, the SDLC link has additional features and modes that are not available in standard ABC type connectors for TS1 Controllers.

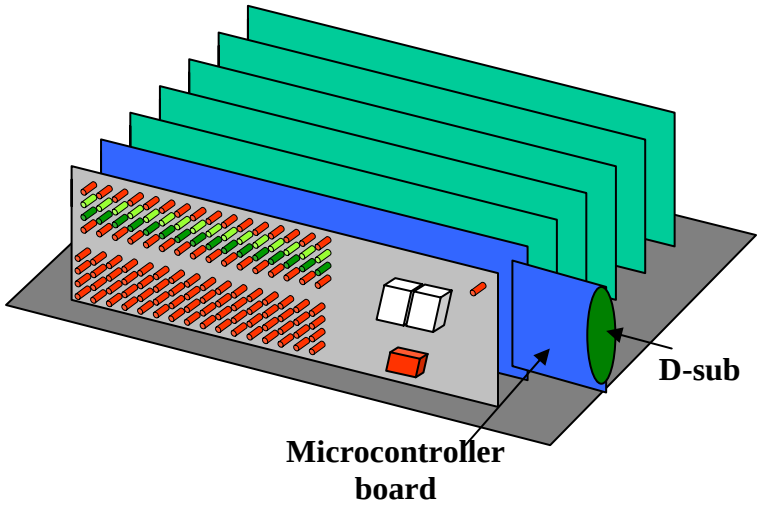

Figure 1a: SDLC microcontroller daughterboard compared to input, output and display boards.

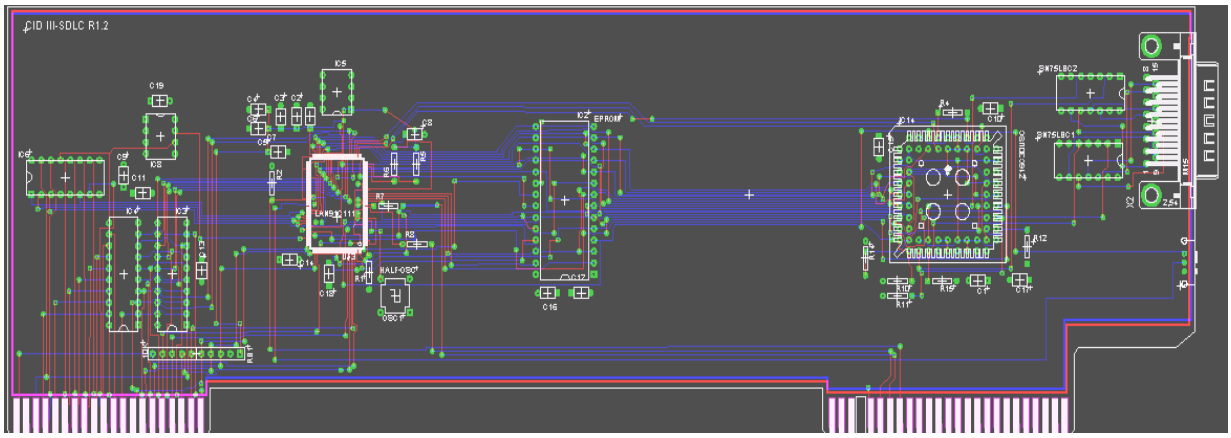

Figure 1b. SDLC CID Circuit Board

B. HARDWARE

1. Revisions

The main revisions include a change to a newer microcontroller; the Cypress FX2. The Cypress EZ- Universal Serial Bus (USB) AN2131 is now obsolete and the FX2 is an upgrade that allows use of the faster USB 2.0 protocol to communicate with the PC.

The second main revision is the inclusion of the Zilog Universal Serial Controller or USC (part number Z16C30) that is used for all SDLC communication processes such as error correction, parity checks, and conversions from parallel to the correct serial bit rate. The integrated circuit (IC) is highly adjustable however the firmware uses only the limited features needed to transfer over the SDLC port.

In order to switch between original and SDLC modes a two position switch is available on the right side of the microcontroller board. For SDLC mode the switch is the down position and for original the switch must be pressed in the upward position.

2. FX2 hardware interface

For proper utilization of the FX2 controller, an external 24 MHz parallel resonant crystal drives the on-chip electronics. For the USB connections, two resistances drop the output and input (USB+, USB-) line voltages. External supply to ground capacitances provide noise reduction to

the IC. External ports are driven from outputs and input pins. A 3.3 voltage regulator converts the 5V supply for proper supply voltage to the FX2 chip. A schematic diagram of this interface is shown in Figure 2.

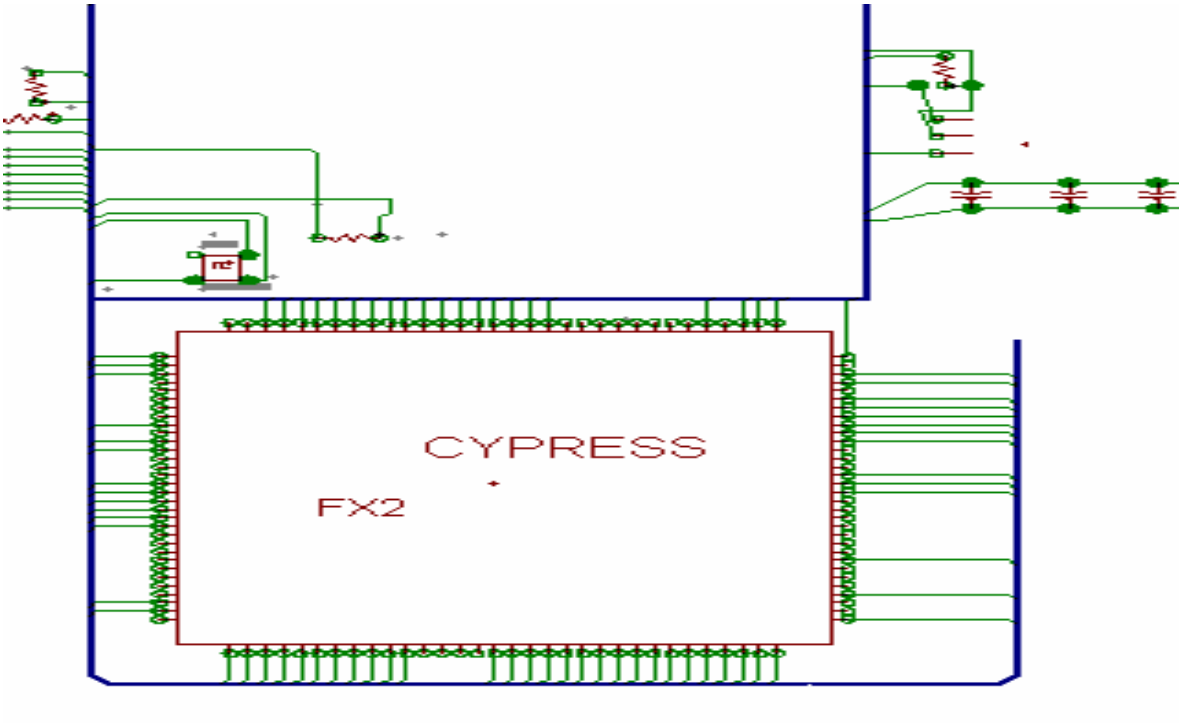

Figure 2. FX2 Hardware Interface

3. SDLC Layout

The USC IC is interfaced to the FX2 through the available address and data buses. Read (RD), Write (WR) and Chip Select Strobes (CS) are carried directly from the FX2 output pins. Power and Ground lines are carried from the 5V power source. Pull-up resistors and Pull down resistors are used on any required floating pins as described in the USC manual.

Table 1 describes the pin connections required in the SDLC processor board. The interface to the USC is done strictly with command and data busses using built in data strobes for chip select, read and write functions. A schematic diagram of the SDLC circuit is shown in Fig. 3.

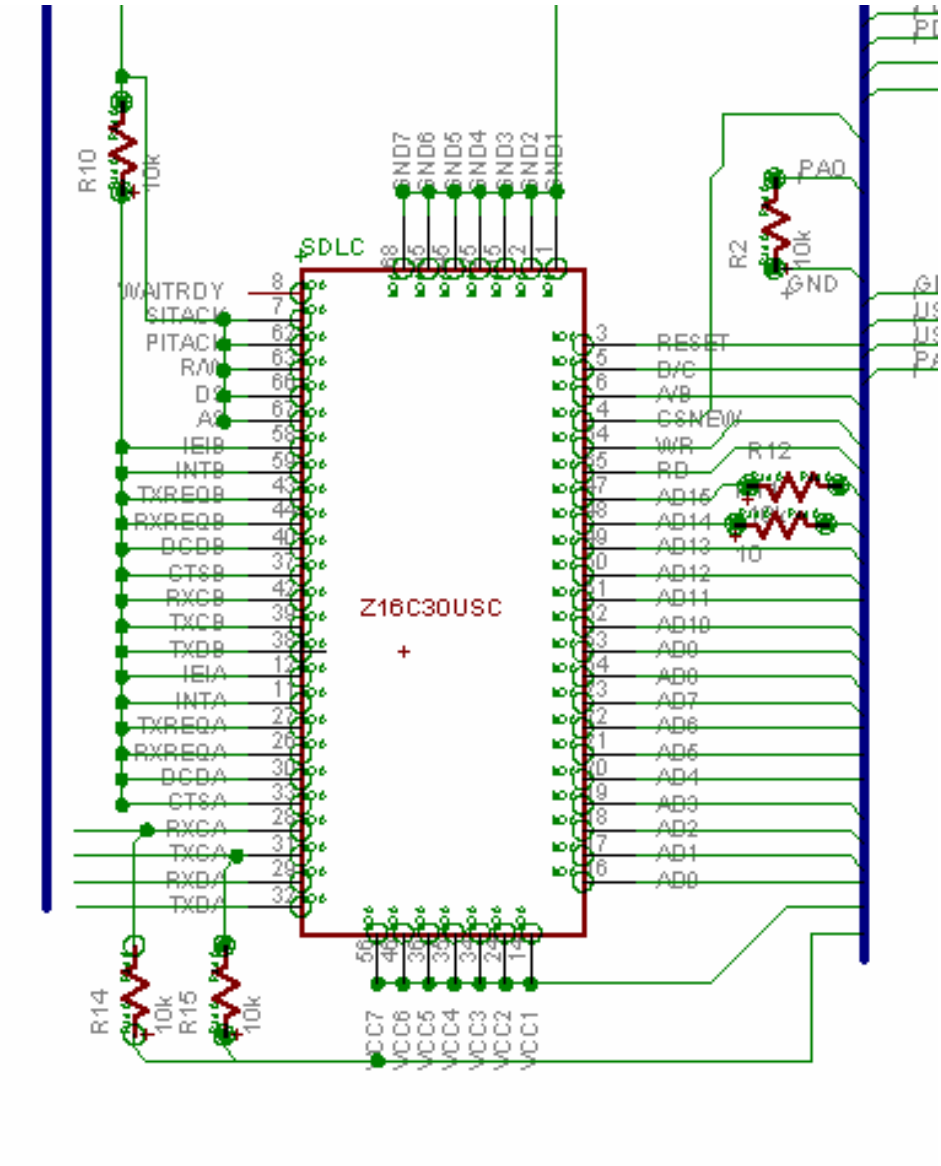

Figure 3. USC Layout

Table 1. USC Pin Interface

```
SDLC PINOUT LAYOUT
ZILOG USC
 *************
 D0 |-------| AD0 - Data 0
 D1 |-------| AD1 - Data 1
 D2 |-------| AD2 - Data 2
 D3 |-------| AD3 - Data 3
 D4 |-------| AD4 - Data 4
 D5 |-------| AD5 - Data 5
 D6 |-------| AD6 - Data 6
 D7 |-------| AD7 - Data 7
 A0 |-------| AD8 - U/L Selector 1 sets upper LSB
 A1 |-------| AD9 - USC Address0
 A2 |-------| AD10 - USC Address1
 A3 |-------| AD11 - USC Address2
 A4 |-------| AD12 - USC Address3
 A5 |-------| AD13 - USC Address5
 A6 |-------| D/C -Sets Data or Control
 A7 |-------| A/B -Sets Channel A - Default Channel A
 RD |-------| RD -Read
 WR |-------| WR -Write
 CS |-------| CS -Chip Select, Activated high for Bus Transfers
 A15 |-------| CSnew -Chip Select, option2
 ***************
```

4. Parts List

A complete list of the parts needed for the SDLC CID microcontroller board is shown in Table 2 along with estimated prices and suppliers.

Table 2. SDLC CID Parts List

Part	Manufacturer	Product Codes		Price
Circuit Board	QTC Circuits			$ 50.00
Cypress EZ-USB microcontroller	Cypress	AN2131QC	428-1307-ND	$ 10.53
12 MHz Clock	ECE Inc.	OECS-2200B-120	XC269-ND1	$ 2.64
3.3 V Voltage Regulator	Linear Technology	LT1121CN8-3.3	LT1121CN8-3.3-ND1	$ 2.75
3 to 8 decoder	Philips Electronics	74HCT238		$ -
Octal Transceiver	Philips Electronics	74HCT245		$ -
EPROM 8X 32k	ST Microelectronics	M27C256B-100DC		$ -
Octal 3 state Transceiver	Texas Instruments	SN74HC245N	296-1584-5-ND	$ 0.53
D-Type Transparent Latch	Texas Instruments	SN74HC245N	296-1596-5-ND	$ 0.53
D Flip Flop with 3 state	Texas Instruments	SN74HC245N	296-1598-5-ND	$ 0.53
USC	Zilog			$ -
RS485 Transscievers	Texas Instruments	SN75LBC180AN	296-6881-5-ND	$ 2.36
Conn PLCC Socket 68 Pos. thru Hole		940-99-068-24-000000	ED80026-ND	$ 0.91

C. FIRMWARE

The firmware for the SDLC CID is modified from the original CIDII firmware with changes to convert to the FX2 IC and additional code to interface with the USC transceiver.

The source code file USC.c contains necessary functions to write, read and initialize the USC chip in the desired mode described in the National Electric Manufacturers Association (NEMA) TS2 standard documentation. USC.h contains all USC bit descriptions and constants needed to set up and communicate with the serial controller.

Table 3. USC Mode Settings

```
BYTE CMRMODE_L = 0x06
BYTE CMRMODE_H = 0x06
BYTE CCARMODE_L = 0x00
BYTE CCARMODE_H = 0x00
BYTE CCSRMODE_L = 0x00
BYTE CCSRMODE_H = 0x00
BYTE CMCR_L = 0x09
BYTE CMCR_H = 0x00
BYTE RMR_L = 0x02
BYTE RMR_H = 0x00
BYTE TMR_L = 0x02
BYTE TMR_H = 0x00
BYTE IOCR_L = 0x08
BYTE IOCR_H = 0x00;
```

For a description of the microcontroller operation refer to the FX2 manual, which is available in the docs folder as well as online at the following URL: www.keil.com/dd/docs/datashts/**cypress**/**fx2**_trm.pdf.

Table 4 lists the source code needed for the SDLC CID and descriptions of each of the files.

Table 4. Firmware Source Code

Filename	Version	Description
Fw.c	1.1	Basic USB operation
cid.c	1.1	Basic CID operation
cid.h	1.1	CID constants
Dscr.h	1.1	USB Descriptor Table
Misc.c	1.1	Various CID functions
Periph.c	1.1	Serial/USB Communications
Periph.h	1.1	Constants
USC.c	1.1	USC functions
USC.h	1.1	USC constants/addresses

D. SOFTWARE

The CID PC software applications will require several modifications. First, the USB communication mode between the PC and CID will need to changed. The original CID II software suite performed PC to CID communication using isochronous USB communication. However, this mode of does not have sufficient bandwidth for SLDDC operation. Instead, bulk mode transfers are required to perform real-time hardware in the loop simulation. Since the guaranteed timing of the isochronous mode operation is not available, it is important to ensure that there are no other data intensive devices using the USB interface on the PC other than CIDs.

A bulk read of the SDLC USC is done with a 0x00 write. A bulk write to the SDLC USC is done with a 0x01 write, followed by the bits desired to be written as described in the protocol shown in Table 5. The protocol used for transfers between the CID and SDLC is described in Tables 5 and 6.

Second, the CID software applications will require modification to account for the additional inputs and outputs available through the use SDLC communication. This will be most noticable with the suitcase tester application.

E. SET-UP

1. Setting up the TS2 traffic controller is SDLC mode

Follow instructions in traffic controller users manual.

2. Setting up and connecting the SDLC CID

- Remove the original CID microcontroller board from the CID by gently pulling upwards on both edges of the board. Be sure to not touch any electronics while doing so.
- Into the empty slot, insert the SDLC CID microcontroller board. The parts on the board should be visible from the front of the CID and the SDLC CID serial board (15 pin Sub D connector) will be on the right edge of the CID.
- Ensure the CID has attached power cable by checking that the C connector to the traffic controller is connected.

- To check that the new board is connected properly switch on the rear power connection on the CID to on. The front L's on the CID should flash left to right. Now turn the power switch back to the off position. To put the CID into SDLC mode, switch the side SDLC switch on the new board to the down position.

3. Connecting to the PC and hardware testing

- Connect a USB cable to the rear of the CID to an available USB slot on the PC.
- Upon connection, the PC should make its standard enumeration sound and a message on the PC should pop up on the left side of the screen that shows the hardware is now available. If not follow instructions on installing SDLC CID drivers.

4. Run application software

Follow normal CID software operation procedure.

F. CONCLUSION

A final version of the modifications to CID to allow SDLC communication between the CID and the traffic controller has been presented. The hardware, firmware and software design has been described.

Table 5. CID to PC Messages

CID to PC message -- This is the message that updates the PC with the TC outputs. To initiate this transfer, the PC must send the command byte 0x01 via BULK ENDPOINT OUT1 to the CID and then read BULK ENDPOINT IN1 for this message.

Byte #	Bit #	Function
0	0	CID Number Bit 0
	1	CID Number Bit 1
	2	CID Number Bit 2
	3	CID Number Bit 3
	4	CID Number Bit 4
	5	CID Number Bit 5
	6	CID Number Bit 6
	7	CID Number Bit 7
1	0	Load Switch 1 Red +
	1	Load Switch 1 Red -
	2	Load Switch 1 Yellow +
	3	Load Switch 1 Yellow -
	4	Load Switch 1 Green +
	5	Load Switch 1 Green -
	6	Load Switch 2 Red +
	7	Load Switch 2 Red -
2	0	Load Switch 2 Yellow +
	1	Load Switch 2 Yellow -
	2	Load Switch 2 Green +
	3	Load Switch 2 Green -
	4	Load Switch 3 Red +
	5	Load Switch 3 Red -
	6	Load Switch 3 Yellow +
	7	Load Switch 3 Yellow -
3	0	Load Switch 3 Green +
	1	Load Switch 3 Green -
	2	Load Switch 4 Red +
	3	Load Switch 4 Red -
	4	Load Switch 4 Yellow +

Byte #	Bit #	Function
5	0	Load Switch 6 Yellow +
	1	Load Switch 6 Yellow -
	2	Load Switch 6 Green +
	3	Load Switch 6 Green -
	4	Load Switch 7 Red +
	5	Load Switch 7 Red -
	6	Load Switch 7 Yellow +
	7	Load Switch 7 Yellow -
6	0	Load Switch 7 Green +
	1	Load Switch 7 Green -
	2	Load Switch 8 Red +
	3	Load Switch 8 Red -
	4	Load Switch 8 Yellow +
	5	Load Switch 8 Yellow -
	6	Load Switch 8 Green +
	7	Load Switch 8 Green -
7	0	Load Switch 9 Red +
	1	Load Switch 9 Red -
	2	Load Switch 9 Yellow +
	3	Load Switch 9 Yellow -
	4	Load Switch 9 Green +
	5	Load Switch 9 Green -
	6	Load Switch 10 Red +
	7	Load Switch 10 Red -
8	0	Load Switch 10 Yellow +
	1	Load Switch 10 Yellow -
	2	Load Switch 10 Green +
	3	Load Switch 10 Green -
	4	Load Switch 11 Red +

Byte #	Bit #	Function
10	0	Load Switch 13 Red +
	1	Load Switch 13 Red -
	2	Load Switch 13 Yellow +
	3	Load Switch 13 Yellow -
	4	Load Switch 13 Green +
	5	Load Switch 13 Green -
	6	Load Switch 14 Red +
	7	Load Switch 14 Red -
11	0	Load Switch 14 Yellow +
	1	Load Switch 14 Yellow -
	2	Load Switch 14 Green +
	3	Load Switch 14 Green -
	4	Load Switch 15 Red +
	5	Load Switch 15 Red -
	6	Load Switch 15 Yellow +
	7	Load Switch 15 Yellow -
12	0	Load Switch 15 Green +
	1	Load Switch 15 Green -
	2	Load Switch 16 Red +
	3	Load Switch 16 Red -
	4	Load Switch 16 Yellow +
	5	Load Switch 16 Yellow -
	6	Load Switch 16 Green +
	7	Load Switch 16 Green -
13	0	TBC Auxiliary 1
	1	TBC Auxiliary 2
	2	Preempt 1 Status
	3	Preempt 2 Status
	4	0
	5	0
	6	0
	7	0
14	0	TBC Auxiliary 3
	1	Free/Coord Status
	2	Preempt 3 Status
	3	Preempt 4 Status
	4	Preempt 5 Status

Byte #	Bit #	Function
16	0	System Special Function 1
	1	System Special Function 2
	2	System Special Function 3
	3	System Special Function 4
	4	0
	5	0
	6	0
	7	0
17	0	Status Bit A Ring 1
	1	Status Bit B Ring 1
	2	Status Bit C Ring 1
	3	Status Bit A Ring 2
	4	Status Bit B Ring 2
	5	Status Bit C Ring 2
	6	0
	7	0
18	0	Phase 1 Phase On
	1	Phase 2 Phase On
	2	Phase 3 Phase On
	3	Phase 4 Phase On
	4	Phase 5 Phase On
	5	Phase 6 Phase On
	6	Phase 7 Phase On
	7	Phase 8 Phase On
19	0	Phase 1 Phase Next
	1	Phase 2 Phase Next
	2	Phase 3 Phase Next
	3	Phase 4 Phase Next
	4	Phase 5 Phase Next
	5	Phase 6 Phase Next
	6	Phase 7 Phase Next
	7	0
20	0	Phase 8 Phase Next
	1	Phase 1 Check
	2	Phase 2 Check
	3	Phase 3 Check
	4	Phase 4 Check

	5	Load Switch 4 Yellow -
	6	Load Switch 4 Green +
	7	Load Switch 4 Green -
4	0	Load Switch 5 Red +
	1	Load Switch 5 Red -
	2	Load Switch 5 Yellow +
	3	Load Switch 5 Yellow -
	4	Load Switch 5 Green +
	5	Load Switch 5 Green -
	6	Load Switch 6 Red +
	7	Load Switch 6 Red -

	5	Load Switch 11 Red -
	6	Load Switch 11 Yellow +
	7	Load Switch 11 Yellow -
9	0	Load Switch 11 Green +
	1	Load Switch 11 Green -
	2	Load Switch 12 Red +
	3	Load Switch 12 Red -
	4	Load Switch 12 Yellow +
	5	Load Switch 12 Yellow -
	6	Load Switch 12 Green +
	7	Load Switch 12 Green -

	5	Preempt 6 Status
	6	0
	7	0
15	0	Timing Plan A
	1	Timing Plan B
	2	Timing Plan C
	3	Timing Plan D
	4	Offset 1
	5	Offset 2
	6	Offset 3
	7	Automatic Flash Status

	5	Phase 5 Check
	6	Phase 6 Check
	7	Phase 7 Check
21	0	Phase 8 Check
	1	0
	2	0
	3	0
	4	0
	5	0
	6	0
	7	0

Table 6. PC To CID Messages

PC to CID Message -- This message is sent to the CID via a bulk transfer on Endpoint 1. A command byte of 0 is the first byte in this message to let the CID know that this is an input update. This makes the total size 25 bytes for this message.

Byte	Bit #	Function
0	0	0
	1	0
	2	0
	3	0
	4	0
	5	0
	6	0

Byte	Bit #	Function
5	0	Detector 33 Call Status
	1	Detector 34 Call Status
	2	Detector 35 Call Status
	3	Detector 36 Call Status
	4	Detector 37 Call Status
	5	Detector 38 Call Status
	6	Detector 39 Call Status

Byte	Bit #	Function
10	0	Test B
	1	Automatic Flash
	2	Dimming Enable
	3	Manual Control Enable
	4	Interval Advance
	5	External Minimum Recall
	6	External Start
	7	TBC On Line
11	0	Stop Time Ring 1
	1	Stop Time Ring 2
	2	Max II Selection Ring 1
	3	Max II Selection Ring 2
	4	Force Off Ring 1
	5	Force Off Ring 2
	6	Call to NA 1

Byte #	Bit #	Function
16	0	Pedestrian Detector 5
	1	Pedestrian Detector 6
	2	Pedestrian Detector 7
	3	Pedestrian Detector 8
	4	0
	5	0
	6	0
	7	0
17	0	0
	1	0
	2	0
	3	0
	4	0
	5	0
	6	Red Rest Ring 1

Byte #	Bit #	Function
22	0	0
	1	Address Bit 0
	2	Address Bit 1
	3	Address Bit 2
	4	Address Bit 3
	5	Address Bit 4
	6	0
	7	0
23	0	Phase 1 Pedestrian Omit
	1	Phase 2 Pedestrian Omit
	2	Phase 3 Pedestrian Omit
	3	Phase 4 Pedestrian Omit
	4	Phase 5 Pedestrian Omit
	5	Phase 6 Pedestrian Omit
	6	Phase 7 Pedestrian Omit

	7	0
1	0	Detector 1 Call Status
	1	Detector 2 Call Status
	2	Detector 3 Call Status
	3	Detector 4 Call Status
	4	Detector 5 Call Status
	5	Detector 6 Call Status
	6	Detector 7 Call Status
	7	Detector 8 Call Status
2	0	Detector 9 Call Status
	1	Detector 10 Call Status
	2	Detector 11 Call Status
	3	Detector 12 Call Status
	4	Detector 13 Call Status
	5	Detector 14 Call Status
	6	Detector 15 Call Status
	7	Detector 16 Call Status
3	0	Detector 17 Call Status
	1	Detector 18 Call Status
	2	Detector 19 Call Status
	3	Detector 20 Call Status
	4	Detector 21 Call Status

	7	Detector 40 Call Status
6	0	Detector 41 Call Status
	1	Detector 42 Call Status
	2	Detector 43 Call Status
	3	Detector 44 Call Status
	4	Detector 45 Call Status
	5	Detector 46 Call Status
	6	Detector 47 Call Status
	7	Detector 48 Call Status
7	0	Detector 49 Call Status
	1	Detector 50 Call Status
	2	Detector 51 Call Status
	3	Detector 52 Call Status
	4	Detector 53 Call Status
	5	Detector 54 Call Status
	6	Detector 55 Call Status
	7	Detector 56 Call Status
8	0	Detector 57 Call Status
	1	Detector 58 Call Status
	2	Detector 59 Call Status
	3	Detector 60 Call Status
	4	Detector 61 Call Status

	7	Walk Rest Modifier
12	0	Pedestrian Detector 1
	1	Pedestrian Detector 2
	2	Pedestrian Detector 3
	3	Pedestrian Detector 4
	4	0
	5	0
	6	0
	7	0
13	0	0
	1	0
	2	0
	3	0
	4	0
	5	0
	6	0
	7	Preempt Detector 3
14	0	Preempt Detector 4
	1	Preempt Detector 5
	2	Preempt Detector 6
	3	Call to NA II
	4	0

	7	Red Rest Ring 2
18	0	Omit Red Clear Ring 1
	1	Omit Red Clear Ring 2
	2	Pedestrian Recycle Ring 1
	3	Pedestrian Recycle Ring 2
	4	Alternate Sequence A
	5	Alternate Sequence B
	6	Alternate Sequence C
	7	Alternate Sequence D
19	0	Phase 1 Omit
	1	Phase 2 Omit
	2	Phase 3 Omit
	3	Phase 4 Omit
	4	Phase 5 Omit
	5	Phase 6 Omit
	6	Phase 7 Omit
	7	Phase 8 Omit
20	0	Phase 1 Hold
	1	Phase 2 Hold
	2	Phase 3 Hold
	3	Phase 4 Hold
	4	Phase 5 Hold

	7	Phase 8 Pedestrian Omit
24	0	Offset 1
	1	Offset 2
	2	Offset 3
	3	
	4	0
	5	0
	6	0
	7	0

	5	Detector 22 Call Status
	6	Detector 23 Call Status
	7	Detector 24 Call Status
4	0	Detector 25 Call Status
	1	Detector 26 Call Status
	2	Detector 27 Call Status
	3	Detector 28 Call Status
	4	Detector 29 Call Status
	5	Detector 30 Call Status
	6	Detector 31 Call Status
	7	Detector 32 Call Status

	5	Detector 62 Call Status
	6	Detector 63 Call Status
	7	Detector 64 Call Status
9	0	0
	1	0
	2	0
	3	0
	4	0
	5	Preempt Detector 1
	6	Preempt Detector 2
	7	Test A

	5	0
	6	0
	7	0
15	0	Inhibit Max Term Ring 1
	1	Inhibit Max Term Ring 2
	2	Local Flash Status
	3	MMU Flash Status
	4	Alarm 1
	5	Alarm 2
	6	Free (No Coord)
	7	Test C

	5	Phase 6 Hold
	6	Phase 7 Hold
	7	Phase 8 Hold
21	0	Timing Plan A
	1	Timing Plan B
	2	Timing Plan C
	3	Timing Plan D
	4	0
	5	0
	6	0
	7	0

PART 2: REAL-TIME PLAYBACK HARDWARE-IN-THE-LOOP SIMULATION OF TRAFFIC SYSTEMS

A. INTRODUCTION *

Traffic signals are controlled by traffic controllers, embedded computers that set light scheduling according to a programmed algorithm. Traffic controllers vary widely in intelligence; they may implement simple fixed-time scheduling systems, more complex traffic-actuated control, or advanced interconnected control systems.

A traffic controller's control outputs are called phase indications; they show allowed movement for vehicles or pedestrians in a certain path. Traffic-actuated controllers receive inputs from traffic detection sensors (usually inductive loops) and from pedestrian call buttons. Phase decisions in an actuated controller are made based on metrics extracted from the input data, including the presence of waiting vehicles, vehicle speed, and traffic volume or density [1].

Traffic engineers frequently use computer "microscopic simulation" tools to design and tune traffic systems. A micro-simulator is a software program that models the behavior of individual vehicles in the system. Common simulators include CORSIM (CORridor SIMmulation), developed by the Federal Highway Administration as part of its Traffic Software Integrated Systems Package [2], and VISSIM (a German acronym; the name means roughly "traffic in towns simulation"), developed commercially by Innovative Transportation Concepts, Inc. [3]; other commercial simulators are also available, but are less widely used at present. These simulators typically provide measures of effectiveness (MOE) for the simulated system, such as total vehicle delay, stopped delay, and queue lengths; detailed run results; and an animation of the system as it is being simulated. Simulations are based on stochastic vehicle models, but are repeatable for a given random seed.

---

*This section consists of the following paper that was presented at the 32nd Annual Conference of the IEEE Industrial Electronics Society., November 6-10, 2005: E. M. Suwal, B. K. Johnson, H. L. Hess, and J. C. Fisher, "Real-Time Playback Hardware-in-the-Loop Simulation of Traffic Systems," *IECON 2005,* pp. 383-388.

B. LIMITATIONS IN MICROSCOPIC SIMULATIONS

Traffic signals and traffic controllers are devices used to control and regulate the flow of traffic at intersections. Optimal traffic signal timing is developed and tested through a variety of traffic optimization and simulation models that simulate the traffic behavior and emulate the possible actions of the traffic controller. With advances in traffic controller computing power and control logic, the issue of whether the generic simulation model controller accurately emulates the actual performance of the field controller has cast considerable doubt on the output of the simulation models. Continuing changes in the control algorithms used in the traffic controllers and the propriety maintained by their manufacturers limit the accuracy of device-specific models [4, 5].

1. Hardware-in-the-Loop Simulation

In a typical simulation, software such as CORSIM simulates a real-world traffic network by moving individual vehicles across a combined surface street and freeway network using accepted vehicle and driver behavior models and simulating various traffic control devices. The software contains algorithms to both track vehicles through a prescribed highway network and to implement a coordinated actuated signal system [6].

Hardware-in-the-loop simulation (HILS) is different in that, instead of having CORSIM simulate controller features, the CORSIM traffic model only simulates the vehicle detector signals. The control strategy is run on an actual traffic controller that will be used in the field. A controller interface device (CID) provides the real-time linkage between CORSIM and the traffic signal controller as shown in Fig. 1 [4, 5, 6]. The CID makes hardware in the loop simulation possible [1].

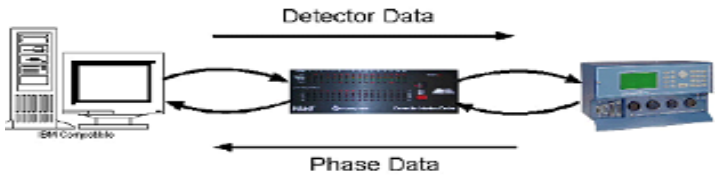

Figure 1: Hardware-in-the-loop simulation with a CID.

The CID is typically an embedded controller that relays detector information from the simulation software to the traffic controller, and returns phase information from the traffic controller to the

simulation software. In this paper, we examine the accuracy of HILS simulations performed using the "CID II," a USB-based interface device developed at the University of Idaho; the methodology is valid for all similar devices.

2. CID Communication Protocol and Latencies

The USB protocol allows up to 127 devices to be connected to a personal computer. Several different data transfer modes are provided to support different types of devices. The CID uses the isochronous transfer mode, which guarantees bounded transfer latency [7].

USB transfers occur in one millisecond long "frames." This provides a very convenient timing reference for the CID II. According to specification, it should be possible to communicate with around 40 CIDs in a single frame using the isochronous transfer method. In practice, it is difficult, if not impossible, to do so. The available USB driver can only write data to or read data from one device per function call, and both read and write calls actually require 6 ms to execute on the computer [5]. The driver can average one transfer per frame, if it is passed a number of packets of data for a particular device, but this is not useful for the purpose of hardware-in-the-loop simulations, in which there is a relatively large time gap between each packet.

In a simulation with only a few CIDs, this should be insignificant, since the simulation time step is usually 1000 ms long. However, in simulations with tens of CIDs, this delay could approach the size of the time-step.

In general, a one time-step timing error does not seem significant; in most simulation systems, time step frequency is chosen well above the maximum transient frequencies. However, there is doubt as to whether a 1000 ms time step length is small enough for advanced traffic control systems; some commercial simulators are moving towards either reducing the size of the time step (for instance to 100 ms) or allowing the user to set the step size.

The two most relevant studies of CID timing issues were undertaken at Louisiana State University with a different type of CID [4], and jointly at the University of Idaho and Purdue University with NIATT's CID II [5]. Results (MOEs) from a number of hardware-in-the-loop simulations for both fixed-time and traffic-actuated controllers were compared to results from

"normal," software-only, simulations and found to have no statistically significant deviation. However, this type of study is not as satisfactory in general as might be hoped: it can only compare results for traffic controllers that can be adequately modeled in software; in fact, there is little need to use HIL simulation with such controllers. Because there is by definition no easy way to model the operation of traffic controllers with proprietary or highly complex algorithms, this evaluation method cannot determine the impact of the CID interface on them.

3. Real-Time Playback

Real-time playback (RTPB) is a discrete-time simulation technique developed for systems in which it is difficult or impossible to "close the loop" between computer simulation and hardware testing—for instance, if the simulation is unable to be run in real time. RTPB simulators have been used by the electric power industry for some time. They provide a cheap alternative to real-time digital simulators (which can cost hundreds of thousands of dollars), in which a simulation actually interacts with the tested system in real-time. RTPB simulators are used principally for testing numerical relays [8], but they have also been used for testing other types of hardware (for instance, fault locators [9]) with fast response times that preclude analog testing. More generally, RTPB can be used to create a quasi-hardware-in-the-loop simulation that is real-time to the limits of a playback device. Interaction with the physical system is independent of both simulation speed and communication latency between the simulator and the hardware.

The simulation procedure is as follows:

1. The initial state of the physical system's outputs is read from the playback device.
2. The computer simulator is started and run for a fixed amount of time, with the *simulated* system's output in the simulation fixed at its initial state. The simulation's inputs to the system are recorded.
3. The simulated system inputs are transmitted to the playback device, and "played" in real-time until a change is observed in the system's outputs.
4. The computer simulator is run again for a fixed amount of time past the previously observed output change. The output change is added to the simulated system's behavior. The simulation's inputs to the system are recorded.

Steps 3 and 4 are repeated, acquiring a new system output change each time, as long as desired. Fig. 2 provides a graphical depiction of the process. The simulator generates inputs to be applied to the physical system. When new output events are found from the playback process, the simulator must be invoked again to generate a new sequence of inputs given the changed system-state. Of course, the system must be in the same internal state at the beginning of each playback run to insure consistency from one playback run to another.

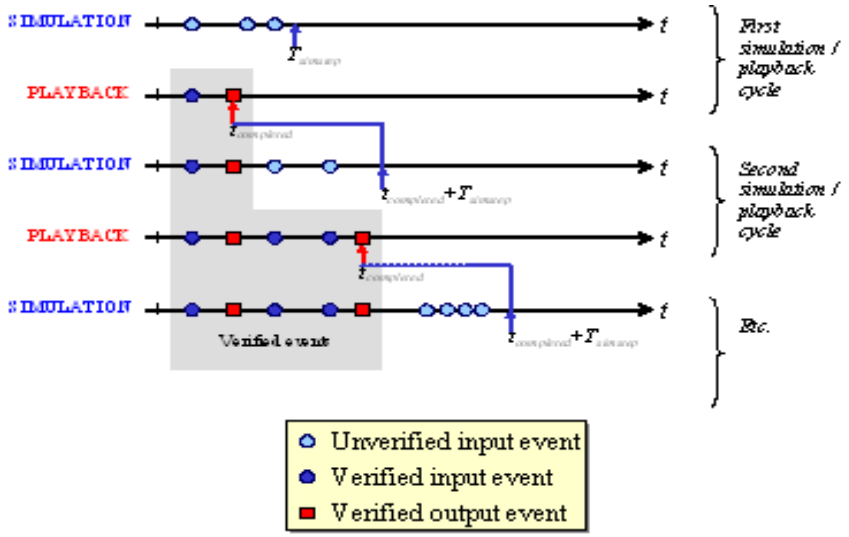

Figure 2. Real-time playback

4. Implementation

The RTPB simulator makes use of the existing CID II hardware, but with modified firmware. The VISSIM traffic simulator was used because of its capability of running with user-selected time step sizes, which allows testing for the impact of time step size as well as latency. A custom control program written in C++ manages the entire RTPB process.

Four principle modifications were made to the CID firmware:

1. Ordinarily, the simulation produces detector information every second, which includes the timing information. In the RTPB simulation, the timing information is read by the

control program, which then creates a new set of detector information without the timing data every 100 milliseconds. The timing resolution is thus traded-off to keep the size of detector data reasonable.

2. A queuing system was added, allowing the CID to store output data for a number of time steps in the future. This removed the possibility of error in the time step length.
3. The CID was programmed to keep track of input (phase) changes. Specifically, the modified CID can record the value and time of the Nth input change after playback has begun (where N can be set by a command over USB).
4. A four-byte simulation clock, incrementing every millisecond, was added to provide a timing base. Functions were added to enable multiple CIDs to synchronize their clocks prior to simulation.

The CID's output data queue length can be set using the control program. Each element in the queue represents a time step and each time step is 100 ms long. The timing clock is four bytes long, but for convenience the least-significant byte rolls over at 100. This allows up to 4605.9 hours of playback before the clock rolls over.

A global "mode" byte allows the CID to keep track of its state from one start-of-frame interrupt to another. These "modes" allows the CID to function differently to make the playback work efficiently. The "mode" is controlled by command from the RTPB control program.
For RTPB to work correctly, all the CIDs should be synchronized, that is, the timing clock implemented in firmware should be synchronized so that the CID detecting the N^{th} change first can be correctly determined.

As it was mentioned before, all the traffic controllers need to be brought to the same state at the start of every simulation and playback process. To achieve this, a set of predetermined detector outputs are fed to the traffic controllers for a sufficient time period to bring them to a known and stable state.

5. Operation

A control program, running on the simulation computer, handles the processes of running the traffic simulator. It runs the playback operation passing data (phase states and detector pulses)

back and forth, and determining when the simulation has completed or stalled. The program was written in C++ as a Windows console program, meaning that it lacks a graphical interface but can still access Microsoft Windows API functions. Fig. 3 illustrates the structure of the RTPB system.

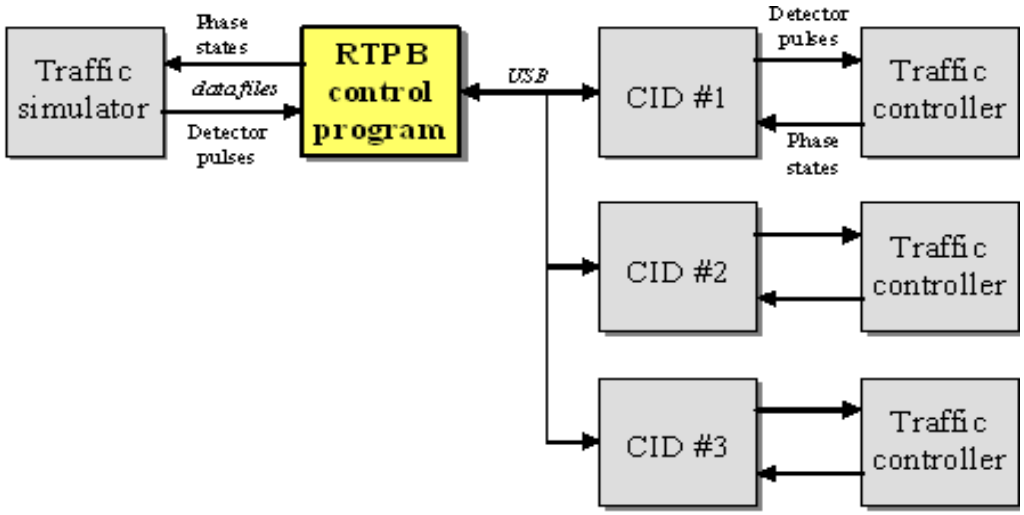

Figure 3. Structure of the RTPB system

In the control program, a C++ object is instantiated for each CID. The class provides methods for sending and receiving data, "shortcut" functions for common CID commands, and data storage arrays for both phase and traffic detector data. When the associated traffic controller's initial phase state is read, it is copied through the CID's entire phase storage array. Then a member function is called to dump the array to a file to be read by VISSIM.

The control program's operation can be divided loosely into three steps: initializing the simulation, running the playback loop, and generating results in a useful format. The control program process is described step by step as follows:

1. The control program first reads a settings file that contains information such as the input file for the simulation, location of the simulator executable file, the number of CIDs that will be used and the length of the playback simulation.

2. The traffic controllers are brought to a known state before starting the simulations. The control program sends out a sequence of detector calls to bring the traffic controllers to the same known state at the start of every simulation.
3. The initial phase states are read from the traffic controllers before starting the simulation. The initial phase states have to be the same every time for RTPB simulation to work.
4. The Simulator is executed using a system command in the control program that automatically loads the input file and starts the simulation, at the end of which, the program is terminated. VISSIM spawns an interface program for each CID that reads phase data from a phase file for the corresponding CID.
5. Before the simulator is terminated, the interface program writes the detector data to detector files for the corresponding CID. The format of the data is the same as that in HILS.
6. The control program processes the detector data before it is sent to the CIDs. Since the timing data is not sent, the resolution of the data is increased for a more accurate playback.
7. The processed detector data is transferred to the corresponding CIDs for playback.
8. The CIDs are synchronized before playback using USB Start-of-Frame (SOF) packet's frame number so that all the CIDs start playback at exactly the same time.
9. The control program then appends newly detected phase states to the phase files for the CIDs.
10. The process is repeated from step 2 until simulation of desired length is completed.

6. Results

Table 1 Comparison of HILS and RTPB Simulation Output Data

HILS			RTPB		
Time of change	SCJ	Phase color	Time of change	SCJ	Phase color
1	8	green	1	8	green
1	7	green	1	7	green
1	6	green	1	6	green
1	5	green	1	5	green
1	4	green	1	4	green
1	3	green	1	3	green
1	2	green	1	2	green
1	1	green	1	1	green
9	7	amber	9	7	amber
12	7	red	12	7	red
14	7	green	14	7	green
16	3	amber	16	3	amber

The RTPB process has been tested with simulations with CID based controllers at one, three and eight intersections. VISSIM can generate an output file recording all the phase changes in the intersections and the simulation time when the phase change occurs. This output file was used to compare the results of hardware-in-the-loop and RTPB simulations. A portion of the output file for an eight-intersection simulation is shown in Table I where SCJ stands for “Signal Controlled Junction.” Since the RTPB simulations take much longer to simulate the same length of simulation than HILS, only 100 seconds of simulation was attempted for the tests. With one- and three- intersection simulations, the results of HILS and RTPB were found to be identical for the length of simulation. For eight-intersection simulation, few discrepancies were observed between RTPB and HIL simulations. The discrepancies were mostly related to one signal controlled junction and are due to the CID in the RTPB process detecting the phase change on that junction at the wrong time. For example, in one of the test simulations, SCJ 4 detects phase change from green to amber after 48 seconds into the simulation and a phase change from amber to red 54 seconds into the simulation. This is incorrect since the traffic controllers are setup with a fixed amber hold time of 3 seconds. The correct change would have occurred after 51 seconds of simulation. The source of this error has yet to be determined but is most likely in the way the firmware clock is implemented.

B. CONCLUSION

A procedure to run hardware-in-the-loop simulation without the affect of communication latency has been designed. This procedure can be used to test the accuracy of hardware-in-the-loop simulations and test the effects of time-step size of the simulation, either in addition to or independent of communication latency. Use of the RTPB process has shown that the HILS process is not creating artificial timing errors in the simulation results.

Acknowledgment

The authors would like to thank Zhen Li for modifying the CID interface software for RTPB.

C. REFERENCES

1. Y. Zhou, "Real-Time Traffic Simulation," MSEE thesis, University of Idaho, 2000.
2. Federal Highway Administration, *Traffic Software Integrated System 97 User's Guide – CORSIM Version 1.03*, New York, June 1997.
3. ITC. (2003). Innovative Transportation Concepts Inc, 1128 NE 2nd St., Ste. 204 Corvallis, OR 97330. http://www.itc-world.com/ (Accessed: April 25, 2005)
4. Bullock, D., and A. Catarella, "A Real-Time Simulation Environment for Evaluating Traffic Signal Systems". Paper presented at the *77th Annual Transportation. Research Board Meeting*. Washington D.C., January 1998.
5. Bullock, D., B. Johnson, R. Wells, M. Kyte, and Z. Li, "Hardware-in-the-Loop Simulation," *Transportation Research Part C: Emerging Technologies*. Vol. 12, Issue 1, pp. 73-89, February 2004.
6. R. L. Gordon, R. A. Reiss, H. Haenel, E. R. Case, R. L. French, A. Mohaddes, and R. Wolcott, *Traffic Control Systems Handbook*. Ch 4. FHWA-SA-95-032, February 1996,
7. USB Documents, "USB 1.1 Specification," http://www.usb.org/developers/docs (Accessed: April 25, 2005).
8. M. S. Sachdev, T. S. Sidhu, and P. G. McLaren, "Issues and Opportunities for Testing Numerical Relays," *IEEE Power Engineering Society Summer Meeting*, July 2000.
9. R. Das, M. S. Sachdev, T. S. Sidhu, "A Fault Locator for Radial Subtransmission and Distribution Lines," *IEEE Power Engineering Society Summer Meeting*, July 2000.