

# A Wideband Distributed Data Acquisition System for High Voltage Applications



K. Ladewig

Thesis presented in partial fulfilment of the requirements for the degree of  
Master of Engineering at the University of Stellenbosch

Supervisors: Dr. H.J. Vermeulen  
Dr. P.J. Bakkes

December 1999

## Declaration

*I, the undersigned, hereby declare that the work contained in the thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.*

K. LADEWIG

25/11/99

DATE

## Opsomming

Hierdie tesis beskryf 'n wyeband dataversamelingstelsel wat geoptimeer is vir wyebandmetings in hoogspanningsomgewings. 'n Kort agtergrond word gegee oor die verskeidenheid toepassings waar so 'n stelsel gebruik kan word. Daar word ook 'n vergelyking getref tussen die ontwikkelde stelsel en die toerusting wat beskikbaar is op die kommersiële mark wat andersins gebruik word vir hierdie toepassings.

Die dataversamelingstelsel het 'n verspreide topologie in dié opsig dat 'n versyferende probe reg by die hoogspanningsomsetter geïnstalleer word. Die versyferde data word dan via hoëspoed optiese vesel serieel na 'n intelligente ontvangseenheid gestuur wat ook buite die substasie geleë is. Die ontvangseenheid kan tot twee kanale akkommodeer en word beheer vanaf 'n persoonlike rekenaar, tipies geleë in 'n substasie, via 'n RS-232 verbinding.

Die stelsel is gebou en daarna getoets in 'n laboratorium. Die stelsel is beheer met behulp van 'n sagteware program wat ontwikkel is. Verskeie analoog seine is versyfer en gestoor met die stelsel en afgelaai na 'n rekenaar. Die stelsel se verrigting asook die geldigheid van die versamelde data is geanaliseer tydens hierdie proses.

Die ontwikkelde stelsel is relatief goedkoop en programmeerbaar, wat dit geskik maak vir 'n groot verskeidenheid toepassings beide in industriële- asook in navorsingsomgewings.

## Abstract

This thesis describes a data acquisition system topology that has been optimised for measuring wide band signals in high voltage (HV) environments. A brief background is given on the various high voltage applications where a wide band acquisition system would be useful. A comparison is made between the developed system and the available commercial equipment that would otherwise be used for these applications.

The topology of the acquisition system is of a distributed nature in the sense that a digitising probe will be implemented close to an HV transducer. The digitised data is then transmitted serially via a high-speed optical fibre link to an intelligent acquisition unit, which is also situated in the external substation environment. The acquisition unit accommodates up to two channels and is controlled via an RS-232 link from a host computer, which will typically be located in the substation.

The data acquisition system was constructed and tested in a laboratory environment. A software control program was developed for controlling the acquisition hardware. A number of analog input signals were captured and downloaded to the host PC. During this process, an analysis was made on the performance and the validity of the acquired data.

The relatively low cost and versatility of the developed system make it suitable for a wide variety of applications in both industrial and research environments.

## Contents

<b>Chapter 1</b>		1
<b>Project motivation and description</b>		
1.1	Introduction	1
1.2	Project motivation	2
1.3	Project description	3
1.4	Overview of thesis	5
<b>Chapter 2</b>		7
<b>Specification and system topology</b>		
2.1	Introduction	7
2.2	Overview of the system specifications	7
2.2.1	Conversion accuracy	8
2.2.2	Sampling rate	9
2.2.3	Storage capacity	12
2.3	Design considerations for the developed acquisition system	12
2.3.1	Earth loops and optical fibre	13
2.3.2	Analog-to-digital converters	14
2.3.3	Amplifiers	15
2.3.4	Multiplexers and sample/hold circuits	15
2.3.5	Anti-aliasing filters	16
2.3.6	Single-ended vs. differential signals	16
2.3.7	First-in, first-out (FIFO) buffers	17
2.3.8	Computer interface	18
	2.3.8.1 Serial ports	18
	2.3.8.2 Parallel ports	18
	2.3.8.3 Direct connection to an internal bus	20
2.4	Topology of the developed system	20
2.4.1	Isolated probes	21
2.4.2	Remote acquisition unit (RAU)	21
2.4.3	Host computer	23

Chapter 3	24
<b>Isolated probes</b>	
3.1 Overview	24
3.2 Analog-to-digital converter (ADC)	25
3.2.1 Analog input signal	26
3.2.2 Track/hold circuit	27
3.2.3 Pipelining and digital error correction	28
3.2.4 Digital output data	29
3.3 Parallel-to-serial encoder	30
3.3.1 Input register	31
3.3.2 Clock generator	31
3.3.3 Encoder	33
3.3.4 Serial outputs	33
3.3.4.1 Positive emitter-coupled logic (PECL)	34
3.3.4.2 PECL output biasing	35
3.4 Fibre optic data links	35
3.5 Fibre optic control links	36
3.5.1 Optical receiver and level translating logic	36
3.6 Control logic	37
3.6.1 Operating modes	38
3.6.2 Executing operating commands	39
3.6.2.1 Acquisition mode	39
3.6.2.2 Idle mode	41
Chapter 4	42
<b>Data receiver module</b>	
4.1 Overview	42
4.2 Optical fibre receivers	43
4.3 Serial to parallel decoders	44
4.3.1 Serial inputs	44
4.3.2 Clock synchronisation	45
4.3.3 Decoding of serial data	46
4.3.4 Implementation	47

4.4	FIFOs/demultiplexers	47
4.4.1	Input control	49
4.4.2	Output control	49
4.4.3	Error situations	50
4.4.4	Implementation	51
Chapter 5		52
<b>System memory module</b>		
5.1	Overview	52
5.2	Dynamic random access memory (DRAM)	53
5.2.1	Read/write cycle	53
5.2.2	Refresh cycle	54
5.2.3	Implementation	54
5.3	RAM interface units (RIU)	55
5.3.1	Writing to DRAM	55
5.3.2	Reading from DRAM	57
5.3.3	Implementation	58
5.4	DRAM controllers	59
5.4.1	Refreshing	60
5.4.2	SIMM configuration logic	60
5.4.3	Address decoding	61
5.4.4	RAS and CAS generators	62
5.5	DRAM address generator	63
Chapter 6		65
<b>RAU control module</b>		
6.1	Introduction	65
6.2	RAU control bus	67
6.3	Implementation and operation of the RAU control module	70
6.3.1	Configuration of the RAU control module	70
6.3.2	Sample clock generator	73
6.3.3	FIFO/demultiplexer control logic	75
6.3.4	RAM interface unit control logic	77

6.3.4.1	Acquisition mode	78
6.3.4.2	Download mode	79
6.3.5	RAM controller and address generator control logic	81
6.3.5.1	Acquisition mode	82
6.3.5.2	Download mode	82
Chapter 7		84
<b>Interface to host computer</b>		
7.1	Introduction	84
7.2	Host interface module	84
7.2.1	Microcontroller	85
7.2.1.1	Functional description of the microcontroller interface	85
7.2.1.2	Implementation	86
7.2.1.3	Microcontroller serial port architecture	86
7.2.1.4	Programming the microcontroller	87
7.2.2	Jobmatch logic analyser	88
7.3	Software environment	89
7.3.1	System control program	90
7.3.1.1	User interface	90
7.3.1.2	Execution of the control program on hardware level	91
7.3.1.2.1	System configuration for <i>idle mode</i>	93
7.3.1.2.2	Acquisition mode	94
7.3.1.2.3	Download mode	95
7.3.1.2.4	Exit	96
7.3.2	<i>Jobmatch</i> software environment	96
Chapter 8		98
<b>Test procedures and results</b>		
8.1	Introduction	98
8.2	Test procedures	98
8.3	Measurements involving the performance of the system components	99
8.3.1	Control signals output by the AT89C2051	99
8.3.2	Control/data signals on the isolated probe	101

8.3.3	Control/data signals to/from the FIFO/demultiplexer	103
8.3.4	Control/data signals to/from the RAM interface unit	105
8.3.5	DRAM controller control signals	108
8.3.6	DRAM address generator control signals	111
8.4	Validation of the captured data	113
Chapter 9		114
<b>Conclusions and recommendations</b>		
9.1	Introduction	115
9.2	Project review	115
9.3	Conclusions	116
9.4	Recommendations	118
<b>Acknowledgements</b>		120
<b>References</b>		121
<b>Appendix A:</b>	CY7B923 data and control codes	A1
<b>Appendix B:</b>	Schematic diagrams for the isolated probes	B1
<b>Appendix C:</b>	Schematic diagrams for the RAU	C1
<b>Appendix D:</b>	VHDL-Code	D1
<b>Appendix E:</b>	Links to sites on programming Altera EPLDs and FPGAs	E1
<b>Appendix F:</b>	Details on 4, 8 and 16 Mbyte SIMMs in system	F1
<b>Appendix G:</b>	AT89C2051 Microcontroller buffers & program	G1
<b>Appendix H:</b>	Deduction of equations of (3.1) and (3.2)	H1
<b>Appendix I:</b>	Software control program code	I1

## List of figures

Figure 1-1	High level block diagram of the developed data acquisition system	4
Figure 2-1	Examples of measured lightning and switching impulse voltages [22]	10
Figure 2-2	A typical corona pulse with its frequency spectrum [22]	10
Figure 2-3	Parial discharge and corona frequency spectra measured in 400 kV substation [1]	11
Figure 2-4	A typical data acquisition system [23]	12
Figure 2-5	An analog input channel	15
Figure 2-6	Analog input sybsystem with multiplexer and sample/hold circuits	15
Figure 2-7	Single-ended vs. differential input configurations	17
Figure 2-8	System topology for the developed acquisition system	21
Figure 2-9	High-level block diagram of the remote acquisition unit (RAU)	22
Figure 3-1	Block diagram of the isolated probe	24
Figure 3-2	ADS801 analog-to-digital converter architecture [13]	25
Figure 3-3	Internal reference structure of the ADS801 [13]	26
Figure 3-4	Input track and hold circuit [13]	27
Figure 3-5	Pipeline architecture and error correction logic [13]	28
Figure 3-6	Timing specifications for the ADS801 output data	30
Figure 3-7	CY7B923 transmitter block diagram [10]	31
Figure 3-8	CY7B923 control signals for transmitting the ADC data	32
Figure 3-9	Buffered PECL switch [15]	34
Figure 3-10	PECL output biasing equivalent [15]	35
Figure 3-11	Optical fibre receiver interface [14]	37
Figure 3-12	Data rate vs. fibre distance for the HFBR-1404/2406	37
Figure 3-13	Block diagram of the control logic module on an isolated probe	38
Figure 3-14	Isolated probe timing diagrams during acquisition mode	40
Figure 4-1	Block diagram of the data receiver module (DRM)	42
Figure 4-2	Block diagram of the CY7B933 receiver logic [10]	44
Figure 4-3	Selection of port B as active differential port for the CY7B933 [15]	45
Figure 4-4	Switching waveforms for the CY7B933 output register	46
Figure 4-5	Block diagram of a FIFO/demultiplexer	48
Figure 5-1	Block diagram of the system memory module (SMM)	52

Figure 5-2	DRAM read and write cycles [29]	53
Figure 5-3	DRAM refresh cycle timing [29]	54
Figure 5-4	Block diagram of a RAM interface unit (RIU)	56
Figure 5-5	Timing waveforms for RIU0 when writing data into the DRAM	57
Figure 5-6	Timing waveforms for RIU0 when reading from the DRAM	58
Figure 5-7	Block diagram of the DRAM controller	59
Figure 5-8	Block diagram of the RAM address generator	63
Figure 6-1	Remote acquisition unit block diagram	66
Figure 6-2	Remote acquisition unit control bus and data bus	68
Figure 6-3	Block diagram of the RAU control module	71
Figure 6-4	Sample clock generator block diagram	74
Figure 6-5	Control logic for the FIFOs/demultiplexers in the RAU control module	75
Figure 6-6	RAM interface unit control in the RAU control module	77
Figure 6-7	RAM interface unit timing waveforms during <i>acquisition mode</i> (both RAM modules active)	78
Figure 6-8	Timing waveforms for the RAM interface unit control logic during <i>download mode</i> (both DRAM modules active)	80
Figure 6-9	RAM controller and address generator control logic in the RAU control module	81
Figure 6-10	Timing waveforms for the DRAM controller and the address generator control logic during <i>download mode</i> (two DRAM modules active)	83
Figure 7-1	RAU host interface module	85
Figure 7-2	Microcontroller interface to the RAU system components	85
Figure 7-3	Flow chart of the AT89C2051 microcontroller program	88
Figure 7-4	User interface for controlling the data acquisition system	90
Figure 7-5	Screen warning if all system parameters have not been set	91
Figure 7-6	<i>Jobmatch</i> software environment	97
Figure 8-1	Testing environment and equipment	99
Figure 8-2	AT89C2051 outputs for <i>idle mode</i> configuration	100
Figure 8-3	Control/data signals measured on the isolated probe during <i>acquisition mode</i>	102

Figure 8-4	Control/data signals to/from the FIFO/demultiplexer during <i>acquisition mode</i>	104
Figure 8-5	Control/data signals to/from the RAM interface unit when writing to DRAM	105
Figure 8-6	Measured control/data signals to/from the RAM interface unit when reading from DRAM	107
Figure 8-7	Control/data signals concerning the DRAM controller during <i>idle mode</i>	108
Figure 8-8	DRAM controller signals when writing to memory	110
Figure 8-9	DRAM controller signals when reading from memory	110
Figure 8-10	Signals concerning DRAM address generator during <i>acquisition mode</i>	112
Figure 8-11	Plot of captured data with 10 kHz sine wave analog input	113

## List of tables

Table 2-1	Harmonic voltage components in a typical EHB system [20]	8
Table 2-2	Parallel port modes [25]	19
Table 3-1	Coding table for the ADS801 [13]	29
Table 5-1	Configuration logic for the DRAM controller	61
Table 5-2	Selection and access to the DRAM SIMMs with the RAS and CAS signals	62
Table 6-1	Op_mode register in the RAU control module	71
Table 6-2	Op_specs register in the RAU control module	72
Table 6-3	Specification of the sample frequency in the RAU control module	72
Table 7-1	Data acquisition system registers to be configured from the host computer	92
Table 7-2	Configuration bytes sent to the microcontroller for <i>idle mode</i> configuration	93
Table 7-3	Configuration bytes sent to the microcontroller for <i>acquisition mode</i> configuration	94
Table 7-4	Configuration bytes sent to the microcontroller for <i>download mode</i> configuration	96
Table 8-1	Measured timing parameters on the isolated probe during <i>acquisition mode</i>	102
Table 8-2	Measured timing parameters concerning the FIFO/demultiplexer during <i>acquisition mode</i>	104
Table 8-3	Measured timing parameters concerning the RAM interface unit when writing to the DRAM	106
Table 8-4	Measured timing parameters concerning the RAM interface unit when reading from the DRAM	107
Table 8-5	Measured timing parameters concerning the DRAM controller during <i>acquisition mode</i> and <i>download mode</i>	111

## List of abbreviations

HV	High voltage
CT	Current transformer
CVT	Capacitive voltage divider
RAU	Remote acquisition unit
EMC	Electro-magnetic compatibility
EMI	Electro-magnetic interference
PECL	Positive emitter-coupled logic
DRM	Data receiving module
SMM	System memory module
A	Ampère
V	Volt
Hz	Hertz
s	seconds
Bd	baud
$\Omega$	ohm
m	milli
k	kilo
n	nano
$\mu$	micro
M	mega
ADC	Analog-to-digital converter
RI	Radio interference
V/F	Voltage to frequency
MUX	Multiplexer
FIFO	First-in, first-out
PC	Personal computer
SPP	Standard parallel port
ECP	Extended capabilities port
EPP	Enhanced parallel port
I/O	Input/output
DMA	Direct memory access

TTL	Transistor-transistor logic
MSB	Most significant bit
LSB	Least significant bit
CM	Common mode
DAC	Digital-to-analog converter
FPGA	Field-programmable gate array
DRAM	Dynamic random access memory
RAM	Random access memory
PCB	Printed circuit board
ECL	Emitter coupled logic
PLL	Phase-locked loop
VHDL	Very high-speed integrated circuit (VHSIC) hardware description language
EPLD	Erasable programmable logic device
RIU	RAM interface unit
SRAM	Static random access memory
SIMM	Single in-line memory module
RAS	Row address strobe
CAS	Column address strobe
AG	Address generator
OE	Output enable
S/P	Serial-to-parallel
P/S	Parallel-to-serial
CS	Chip select
UART	Universal asynchronous receiver/transmitter

# Chapter 1

## Project motivation and description

### 1.1 Introduction

Large electric fields are present in high voltage power system environments. Due to these fields many high frequency noise signals are continually generated and electrically injected into the system. Examples of such phenomena include switching and lightning impulses, corona noise and partial discharge activity. For many years now extensive studies have been done on the removal of these inadvertent signals from control and monitoring systems. However, modern technology is now allowing the research to expand toward the capturing of the noise signals and an analysis of the data. Some promising results have been obtained in this respect. It has been shown that many of the wide band signals, now no longer identified as noise, contain valuable information concerning condition and system monitoring applications [[1], [2], [3], [4]].

Condition monitoring of apparatus in power system environments is becoming a prime focus internationally. Accurate predictions concerning the expected lifetime of apparatus can improve maintenance and prevent major system failures costing millions of rands. Research has shown that a successful method of non-intrusive condition monitoring is the analysis of the high frequency noise originating from substation apparatus [2]. Both the time- and frequency-domain properties of these wide band signals contain information that is used for diagnostic purposes. Examples of such properties include the frequency response signatures that are associated with certain device failures.

In electrical transmission systems the capturing of time- and frequency-domain data of wide band phenomena is also becoming increasingly important. The data is used for analysing the behaviour and performance of the systems as well as individual equipment [1]. For instance, it is known that faults in lines induce wide band noise signals. These signals can be

attenuated and detected with tuned circuits in order to determine the location of the faulty section of the line [[5],[6]].

The equipment currently being used for wide band measurements in high voltage environments is expensive and the measuring procedures are time consuming. Large transducers such as Haefely dividers have to be transported to the measuring site and de-energising of the system is necessary for installation of the transducers. Very expensive digital storage oscilloscopes are used for capturing the data. The research concerning the analysis of the wide band signals has been progressing very slowly due to the high costs and lengthy procedures involved in wide band measurements.

In recent years advanced research has been done on the feasibility of utilising standard substation equipment such as insulated CTs and CVTs as wide band transducers [[7], [8]]. The studies have yielded promising results, opening new doors for the analysis of wide band signals. Now instead of using large mobile transducers for measurements, equipment that is permanently connected in the system can function as the transducers. In this way the cost of wide band measurements can be reduced substantially. The measurements can also be done non-intrusively (de-energising of the system is unnecessary). There is, however, still no replacement for the expensive digital storage oscilloscopes required for any such measurements.

This thesis describes the development of a cost effective high-speed prototype acquisition system for purposes as motivated above. The system is designed for permanent or extended temporary integration into a high voltage substation environment for research and operational purposes. In this chapter further motivation for the project is given, together with a description of the project. In the final section an overview of the thesis layout is presented.

## **1.2 Project motivation**

Research concerning the analysis of the high frequency signals present in high voltage environments for system and condition monitoring purposes yielded promising results. The progress of this research is, however, relatively slow as a result of the expensive equipment and time consuming procedures required for the practical measurements.

The recent discovery that existing substation equipment such as CTs and CVTs can be used as wide band transducers [[7], [8]] has opened new doors for wide band measurements in high voltage environments. With the development of an affordable acquisition system for permanent or temporary integration into the substation, the cost and prolixity of the required measuring procedures can be further reduced.

This combination of standard substation equipment together with a cost-effective acquisition system will enable wide band measurements and tests to be performed quickly and efficiently. Extensive studies can then be conducted on a wide variety of high frequency phenomena, of which the applications include:

- Determining the severity and penetration of harmonic distortion.
- Protection applications, such as monitoring high frequency fault waveforms.
- Carrier frequency applications.
- Monitoring the responses of apparatus and subsystems under transient conditions such as switching and lightning impulses.
- Non-intrusive condition monitoring of power system apparatus.
- Corona and interference studies.

It is clear that there are many applications in high voltage environments where a high-speed data acquisition system can be utilised. With establishment of the specifications of the system it is imperative that these applications be considered.

### **1.3 Project description**

This thesis establishes a data acquisition system topology that is optimised for use in high voltage power system environments. A prototype acquisition system is designed and assembled. Certain design concepts are presented and the performance of currently available technology is demonstrated in order to provide a basis for future work in this field. The development of the prototype system involves the following aspects:

- Determining the required specifications of the system.
- Determining the system topology.
- Design of the system on schematic level.
- Printed circuit board layouts and construction.

- Development of a software control program.
- Testing and evaluation of system in laboratory environment.

Figure 1-1 shows a high-level block diagram representation of the developed system.

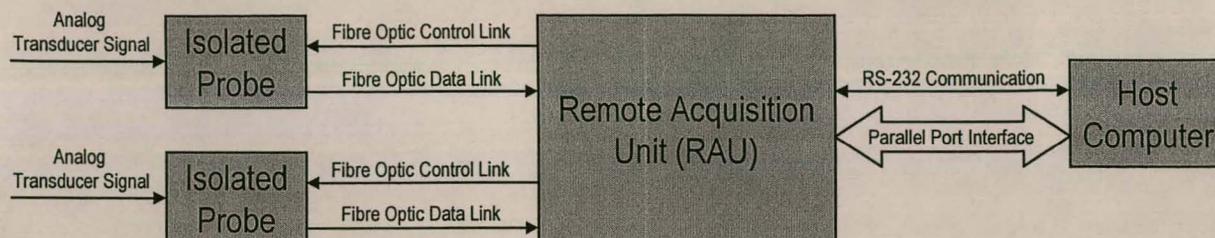


Figure 1-1 High level block diagram of the developed data acquisition system

The topology is of a distributed nature in the sense that two digitising probes are implemented in close vicinity of the high voltage transducers. Each *isolated probe* is controlled via a fibre optic link from a *remote acquisition unit* (RAU). The function of the *isolated probes* is to digitise the transducer signal and to transmit the sampled data to the RAU via the fibre optic control links. The RAU is also situated in the external environment, i.e. external to the substation buildings. In the RAU the data from both probes is stored in real-time in a dedicated memory module.

Operation and configuration of the RAU is performed from a host computer that would typically be situated inside the substation buildings. A software control program enables the user to initiate data acquisition cycles. Data acquisition criteria such as sampling speed, number of active probes, etc. can be programmed from the host computer via an RS-232 link. Once the desired data has been sampled and stored, it is downloaded to the host computer from the RAU for further processing and scrutiny. The parallel port of the computer is used for downloading the data.

The topology of the system as shown in Figure 1-1 allows for a number of specialised features, including:

- Real-time digital signal processing.
- Real-time storage of data for up to two channels.
- Synchronised sampling of the two channels, enabling time stamping of the data.

The distributed nature and optical isolation links in the chosen topology eliminate much of the earthing and EMC problems experienced with multi-channel data acquisition in high voltage environments. The relatively low cost and versatility of the system make it particularly suited for research environments.

#### 1.4 Overview of thesis

The basic specifications for the system were determined by considering the possible applications of the system for high voltage research. Subsequently, the status of available technology, the cost of components and the implications of the technology for design and manufacturing processes were considered. In chapter 2 the procedure for establishing the specifications is described. This chapter also discusses the various functions of each subsystem and motivates the topology chosen for the overall system.

Implementation of an *isolated probe* shown in Figure 1-1 is described in chapter 3. The different components comprising this module are reviewed on block diagram level. The operation of the analog-to-digital converter used is discussed, and some background is given on the logic levels (PECL) at which the optical fibre links operate. Programmable logic devices are implemented as control logic on each probe board. The operation of these devices is reviewed.

The RAU consists of four modules, namely the *data receiving module* (DRM), the *system memory module* (SSM), the *RAU control module* and the *host interface module* (HIM). The data from the *isolated probes* is received by the DSM and saved in real time in the SSM. After acquisition it can be downloaded to the host computer via the *host interface module*. The *RAU control module* generates control signals for the three other modules on the RAU. The DSM, SSM and *RAU control module* are discussed in chapters 4, 5 and 6 respectively. The host interface hardware and software are reviewed in chapter 7.

The developed acquisition system was tested and evaluated in a laboratory environment. The results obtained are presented in chapter 8. The final chapter discusses a number of conclusions that were made on completion of the project. The system was found to comply with the specifications determined in chapter 2. However, further work is required before the

system can be deployed in an actual substation environment. Recommendations for this work are also discussed in chapter 9.

## Chapter 2

### Specifications and system topology

#### 2.1 Introduction

The specifications for the data acquisition system were determined by considering the various applications of the system. These specifications are discussed in section 2.2 of this chapter. Section 2.3 introduces some of the aspects that were considered in determining the topology of the system. The topology that was eventually chosen is described and motivated in section 2.4.

The system has been designed specifically for high voltage environments and this was of primary importance in determining the system topology. Earthing problems and electromagnetic interference are common in high voltage environments and their effects have to be eliminated or minimised by appropriate design methodology. The status of the currently available technology has also imposed certain constraints on the system topology.

#### 2.2 Overview of the system specifications

Various aspects were considered in determining the specifications of the data acquisition system. A synoptic study was done of the signals intended to be measured with the system, looking for instance at time-domain properties, frequency-domain properties, amplitude ranges, etc. The properties of the transducers to be utilised with the system were also reviewed. The following subsections present the various system specifications, i.e. conversion accuracy, sampling speed and storage capacity.

The following applications were considered in determining the system specifications:

- Harmonic penetration studies.
- Carrier frequency applications.
- Impulse measurements.

- Corona, partial discharge and other high frequency noise measurements.

### 2.2.1 Conversion accuracy

One of the possible applications for the developed system is for studying harmonic penetration in power systems. In such cases the different harmonic components of the voltage or current waveforms need to be measured. Table 2-1 [20] shows values for the relative amplitudes of the harmonic components in a typical high voltage system with a line voltage rating of 400 kV.

Table 2-1 Harmonic voltage components in a typical EHV system [20]

Harmonic Order	Amplitude (% of $V_{fund}$ )
1	100.0
3	2.0
5	2.0
7	2.0
9	1.0
11	1.5
13	1.5
15	0.3
17	1.0
19	1.0
21	0.2
23	0.7
25	0.7

$V_{fund}$  = amplitude of fundamental voltage component

It can be seen that in order to measure up to the 25th harmonic component, an amplitude resolution of significantly better than 0.1 % is required.

An amplitude resolution of better than 0.1 % is also required for applications such as the measuring of high frequency noise signals and the measuring of carrier frequency signals superimposed on the power frequency cycle [20].

Lightning and switching impulses play an important role in power system environments. The developed system could be ideal for monitoring the responses of system apparatus and subsystems under these conditions. Several national and industrial laboratories in Europe and the US have been doing intercomparative studies in order to determine the measuring

specifications required by impulse measuring systems. It has been found that if impulse parameters are to be evaluated an amplitude resolution of 0.2 % to 0.4 % is required [21].

Practical analog-to-digital converter technology typically offers resolutions of 8-bits, 10-bits and 12-bits for sampling speeds in the MHz range. ADCs with higher resolution are available, but the conversion speeds tend to decrease as the resolution increases. The amplitude resolution obtainable with the above mentioned ADCs are respectively 0.8 %, 0.2 % and 0.048 % with an additional sign bit. In two of the applications discussed earlier in this section an amplitude resolution of better than 0.1 % is needed. A 12-bit ADC is therefore implemented in the data acquisition system.

### 2.2.2 Sampling rate

In determining the severity and penetration of harmonic distortion in power systems, measurements are usually confined to frequencies below  $50f_{fund}$ , where  $f_{fund}$  is the power frequency. In South Africa the fundamental power frequency is 50 Hz and in America it is 60 Hz. Consequently, for harmonic penetration studies the developed system would require a bandwidth of at least 2.5 kHz in South Africa and 3 kHz in America.

The frequency band that is used for power line communications varies between 3 kHz and 500 kHz in different countries. In South Africa, for instance, the power line carrier frequency band has been set at 50 kHz to 500 kHz [18], whereas in the USA it is between 100 kHz and 450 kHz [19]. In Europe the frequency band is specified between 3 kHz and 148.5 kHz [19]. It follows that a bandwidth of 500 kHz is required in order to utilise the developed system for general carrier frequency applications.

The sampling speed required for capturing lightning and switching impulses can be determined by looking at the rise times of these impulses. Figure 2-1 [22] gives examples of typical lightning and switching impulses. The rise time for a standard lightning impulse is in the order of 1  $\mu$ s. An attempt to capture such an impulse digitally would require an ADC with sample rate of at least a few MHz. The rise time of the switching impulse is much slower and of the order of 150  $\mu$ s [22]. For measuring switching impulses a sample rate of a few hundred kHz would therefore suffice.

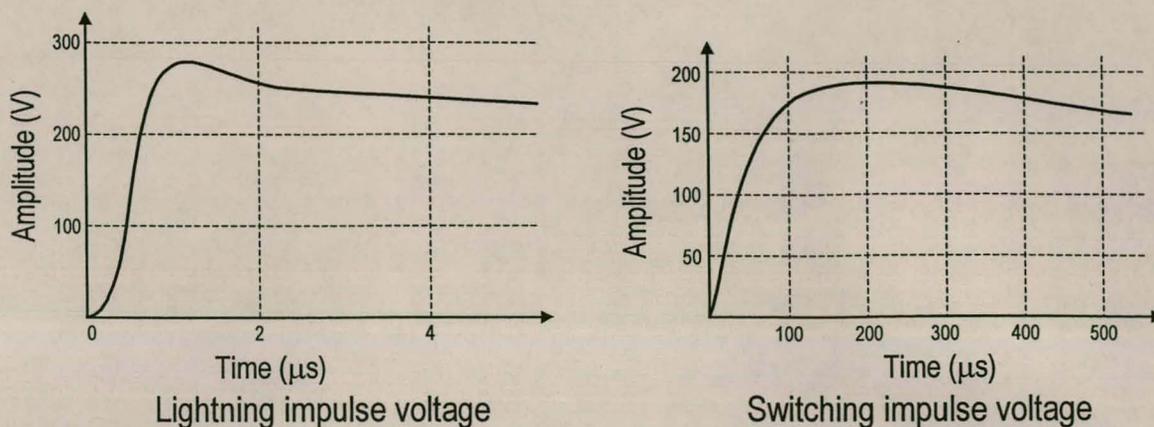


Figure 2-1 Examples of measured lightning and switching impulse voltages [22]

Corona discharge is an important source of radio interference (RI) in power systems environments. These discharges form when the electric field intensity around a conductor exceeds the breakdown strength of air and localised ionisation of the air around the conductor occurs. There are many factors that influence the formation of corona. Figure 2-2 [22] shows the time- and frequency-domain properties of a typical corona pulse.

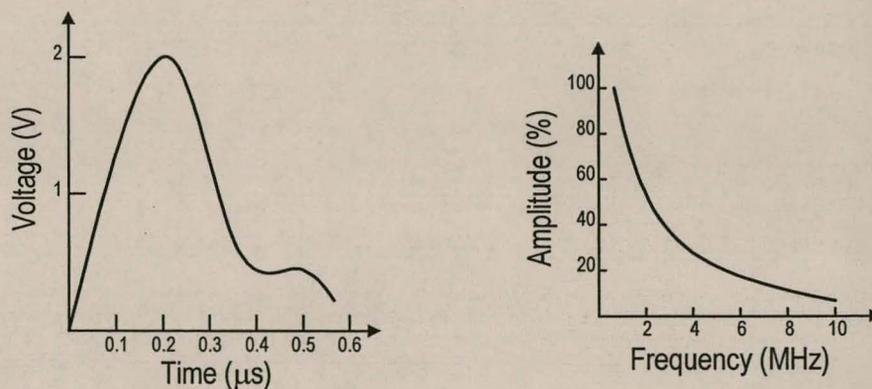


Figure 2-2 A typical corona pulse with its frequency spectrum [22]

In order to record signals such in the time-domain, a sampling rate of the order of tens of MHz would be required. Regarding frequency-domain measurements of corona pulses, the Nyquist theorem must be considered. Frequency components up to 10 MHz are present in a corona pulse, requiring a sampling rate of at least 20 MHz.

Another wide band phenomenon of interest that occurs in high voltage environments is that of partial discharging activity. Partial discharge signals are injected into the network when the insulation in apparatus such as transformers starts to break down. By monitoring these signals, an indication of the condition of power system equipment can be obtained [[16],

[17]]. Partial discharges appear as pulses on the network and typically have a frequency spectrum as shown in Figure 2-3 [1]. Figure 2-3 [1] also includes the frequency spectrum of corona on the dB-scale.

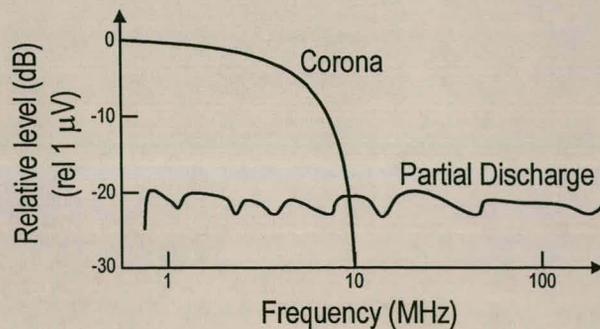


Figure 2-3 Partial discharge and corona frequency spectra measured in 400 kV substation [1]

Figure 2-3 shows that partial discharge has a frequency spectrum that covers the range from below 1 MHz to approximately 100 MHz. In order to detect partial discharges in a noise free environment, a sample rate of a few MHz would suffice. However, in an environment where corona is also present a higher sample rate is required. This can be seen from Figure 2-3. Below 10 MHz the spectral components of corona are much larger than the partial discharge components. Above this frequency the effect of corona disappears and the partial discharge spectral components become measurable.

It was mentioned in chapter 1 that the developed system is intended to be used with standard substation equipment such as CTs and CVTs. In determining the sampling rate of the acquisition system, it is therefore important to also consider the wide band properties of these high voltage transducers. Vermeulen [7] concludes that CTs and CVTs can be used as wide band voltage transducers for frequencies up to 500 kHz. At frequencies higher than 500 kHz the self-resonant frequencies of the devices are approached and they can no longer be used as voltage transducers.

Of all the applications discussed in this section, the frequency components contained in partial discharges and lightning impulses are the highest. In order to measure these types of signals the acquisition system would require a bandwidth in the order of at least 10 MHz. However, the bandwidth obtainable with the CVT and CT transducers is of the order of 500 kHz. For the intended use of the system a bandwidth higher than this would therefore be excessive.

The Nyquist sampling theorem states that, in order to be able to reconstruct a signal, the sampling rate must be at least twice the signal bandwidth. However, to allow for the roll-off of anti-aliasing filters above the 500 kHz cut-off frequency, a sampling rate of 5 to 10 times this frequency is required. This yields a maximum sampling rate of 5 MHz. In addition, the sampling rate that is required for applications such as harmonic distortion measurements, a much lower sampling rate is required. It follows that the sampling rate should be programmable.

### 2.2.3 Storage capacity

The required storage capacity is determined by the conversion resolution, the sampling rate and the data sequence length to be captured by the acquisition system. The conversion resolution and maximum sampling rate have been established at 12 bits and 5 MHz respectively.

The acquisition system has been developed specifically for capturing wide band information superimposed on 50 Hz power signals. If the high frequency data is to be recorded with point of wave information, a complete 50 Hz cycle or 20 ms of data has to be captured. At a sampling rate of  $5 \cdot 10^6$  samples/second, 100 000 samples have to be stored. If each 12-bit sample is stored as two 8-bit bytes, a minimum storage capacity of 200 kbytes per channel is therefore required.

## 2.3 Design considerations for the developed acquisition system

A complete data acquisition system typically consists of the elements shown in Figure 2-4 [23].

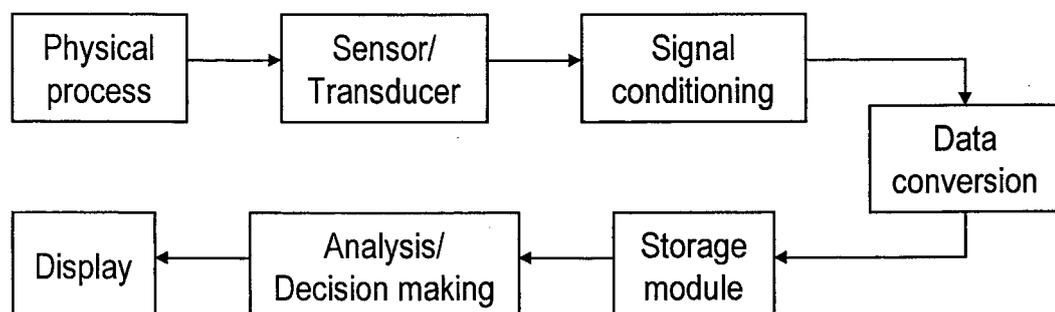


Figure 2-4 A typical data acquisition system [23]

Such a system collects data, which represents a physical phenomenon. A transducer or sensor converts the information contained on the physical process of interest to an electrical signal. Some sensors and transducers then require a signal conditioning circuit. The resulting electrical signal is sampled and converted to digital data by an analog-to-digital converter. The data is saved in the storage module from where it can be recalled for analysis or inspection.

Several data acquisition devices can be distributed around a plant and controlled by one or more supervisory software systems typically hosted on a personal computer. A communication link must then be established between the acquisition devices and the control and acquisition software.

The data acquisition system discussed in this thesis contains only three of the modules shown in Figure 2-4. These are the *data conversion* module, *data storage* module and the *display* module. As was discussed in chapter 1, standard substation equipment such as CTs and CVTs are to be used as wide band voltage transducers with the developed system. The *sensor/transducer* and *signal conditioning* modules in Figure 2-4 are therefore not included in the design of the system. The *analysis/decision-making* module could include features such as digital signal processing, and is discussed in chapter 9 as future work on the system.

When determining the exact configuration of the four elements in Figure 2-4, various aspects were considered. The most important design considerations are reviewed in the following subsections.

### **2.3.1 Earth loops and optical fibre**

One of the more common problems encountered in high voltage environments is the so-called earth loop problem. This arises when multiple sensors with different earth potentials are connected to input circuits that are not isolated from each other. To eliminate earthing problems only data acquisition hardware with inputs which are fully isolated from each other are used.

In the domain of electrical power systems a transmission medium that is becoming very attractive and economically viable is optical fibre. The use of optical fibre has largely been

restricted to the telecommunications industry until recently, but it is now emerging in other fields such as data communications and control applications. The wide bandwidth available and the relative immunity to electromagnetic interference (EMI) are becoming increasingly advantageous. Optical fibre overcomes, for example, the problems of differential earth potentials between locations and is largely immune to lightning strikes. For these reasons optical fibre is used as isolation medium in the acquisition system.

### 2.3.2 Analog-to-digital converters

The type of analog-to-digital converter (ADC) used is an important consideration in the design of the analog input system. There are four common types of ADCs used in data acquisition equipment [11]:

- Voltage-to-frequency (V/F) converters.
- Integrating ADCs.
- Successive approximation ADCs.
- Flash converters.

V/F converters provide very high resolution (16 – 24 bits) and good noise immunity, but their maximum sampling speed is low. Similar performance is obtained with integrating ADCs although they are cheaper than the V/F converters. Successive approximation ADCs are faster than the first two types of converters. The resolution obtained with these ADCs is lower (10 – 16 bits) and they have very little noise immunity. Finally, flash converters achieve very high conversion rates but they are limited to low resolution (4 – 8 bits). Flash converters are also expensive and provide no noise immunity.

The data acquisition system must have a maximum sampling speed in the MHz-range in order to acquire the desired wide band information. For this reason V/F converters and integrating ADCs are not suitable. Successive approximation ADCs are faster than the first two types but although they have good resolution (10 – 16 bits) they are not fast enough for the application. Therefore a type of flash converter is used in the developed system. The specific ADC that is used compensates for the low resolution of flash converters and for their lack in noise immunity. This is discussed further in chapter 3.

### 2.3.3 Amplifiers

A simple analog input channel for data acquisition systems is shown in Figure 2-5.

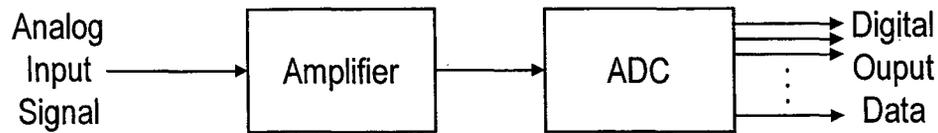


Figure 2-5 An analog input channel

In order to make the best use of the limited resolution of the ADC, the input signal must be amplified so that its range matches the full-scale input range of the ADC. Caution must, however, be taken that the noise generated in the amplifier does not affect the overall performance of the system. In most data acquisition systems the amplifier features a number of selectable gain values and many systems allow the control software to set the gain value (programmable gain amplifier).

No amplification of the analog input signal is done in the developed system. Although this will be an essential part of the system when eventually implemented in a field environment, it is left to be included at a later stage.

### 2.3.4 Multiplexers and sample/hold circuits

Many applications require that two or more channels must be sampled at the same instant. This is usually achieved with the sample/hold system shown in Figure 2-6.

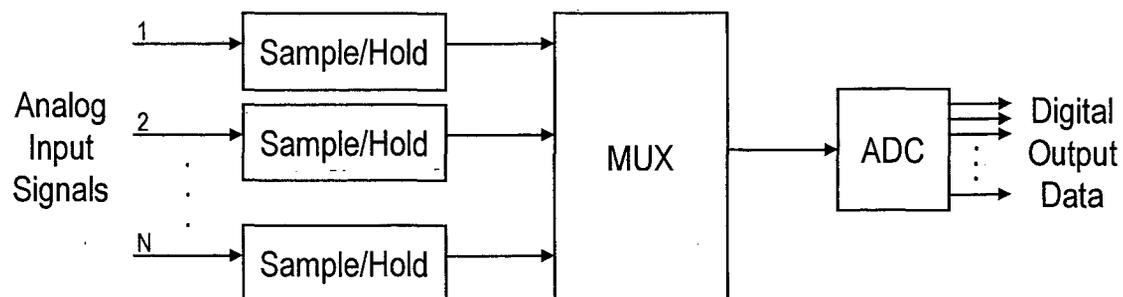


Figure 2-6 Analog input subsystem with multiplexer and sample/hold circuits

When instructed to sample, the inputs on all channels are captured and the signal values are

held until the ADC has sampled each of them. With the use of a multiplexer (MUX) all the channels share a single ADC. The multiplexer switches the sampled values from each sample/hold circuit sequentially to the ADC to be converted. The sample/hold circuits then sample new values and the process is repeated.

The configuration shown in Figure 2-6 is a relatively cheap method to accommodate more than one channel. The data acquisition rate per channel is, however, slowed down because only one ADC is used. Furthermore, the configuration does not allow for the sampling of signals that are physically far apart from each other, or for the inputs to be isolated from each other. As these aspects are of paramount importance in the acquisition system, a separate ADC and optical fibre link is implemented for each channel. In this way the multiple channels are then accommodated. Although it is an expensive solution, it solves the isolation and distributed topology shortcomings of the topology given in Figure 2-6.

### **2.3.5 Anti-aliasing filters**

If a signal to be digitised contains components at a frequency higher than half the sampling frequency, these components appear at a lower frequency in the sampled data. The apparent frequency of a sampled signal is the actual frequency modulo half of the sampling rate. This effect is called aliasing. For all applications involving digital signal processing such as digital filtering, harmonic decomposition, etc., the influence of the higher frequency components have to be eliminated. Anti-aliasing filters are then implemented to cut off all signal components above one half of the sampling frequency before digitisation.

Anti-aliasing filters are omitted in the design of the data acquisition system. This is because no digital signal processing features are implemented in the system in this thesis. These types of filters do, however, have to be considered with eventual implementation of the system in the field, and are recommended for future work in chapter 9.

### **2.3.6 Single-ended vs. differential signals**

In systems where data is taken from a number of different devices that are located long distances apart, large differences can exist between the ground potentials of the devices. The

data acquisition is made possible by inserting a FIFO buffer between the sampling and storage module of the acquisition system.

### **2.3.8 Computer interface**

Computers have become an essential component of data acquisition systems. They are typically used to control systems and for the recording and/or processing of captured data. With modern personal computers (PCs), hardware can be interfaced to the PC by using any of the following:

- The serial port.
- The parallel port.
- A direct connection to the internal bus through a dedicated plug-in circuit board.

#### **2.3.8.1 Serial ports**

Serial ports are general-purpose ports that interface serially to external hardware. In most PCs they conform to the RS-232C or RS-422 standards [24]. When using the serial port, fewer wires are needed than with parallel ports. Furthermore, the serial port can have a maximum voltage swing of 50 V compared to 5 V with a parallel port [24]. Therefore cable loss is not so much of a problem for serial transmission as for parallel transmission and longer cables can be used. The maximum standard data transfer rate through the serial port is specified as 115.2 kbits/s [24].

The serial port of a PC is used in the developed system for controlling the acquisition hardware. The RS-232 standard protocol allows easy interface to most microprocessors. The maximum data transfer rate of 115 kbits/s is, however, too slow for downloading the sampled data.

#### **2.3.8.2 Parallel ports**

The parallel port allows for the input of up to 9 bits or the output of up to 12 bits at any given time. There are a number of modes that the parallel port can be configured in for different applications. These modes are given in Table 2-2 [24].

effect of different ground potentials on both single-ended and differential lines is illustrated in Figure 2-7.

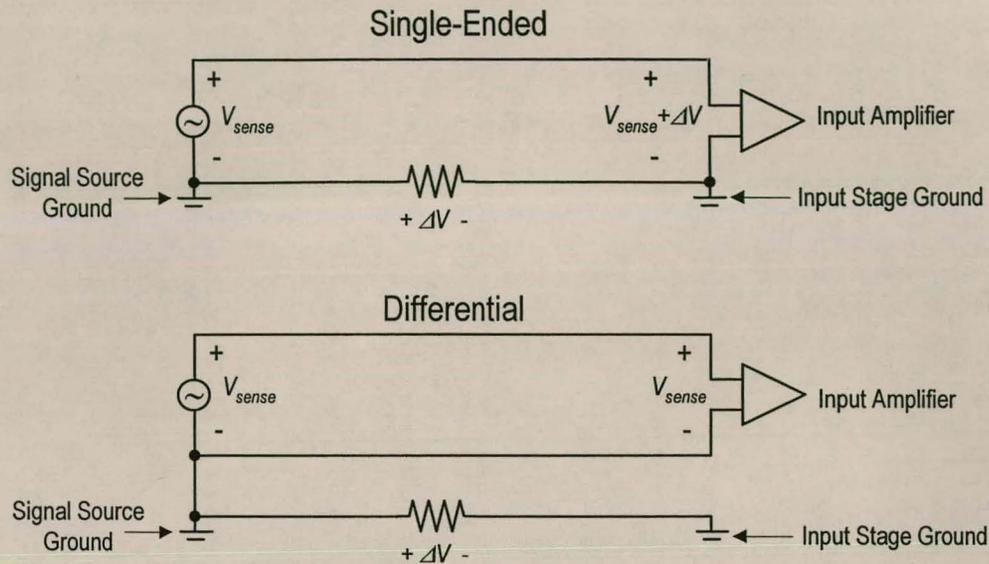


Figure 2-7 Single-ended vs. differential input configurations

Single-ended configurations share a common ground or return line, which connects to the input stage through the system ground connections. The voltage at the signal source ground is, however, not necessarily the same as the input stage ground voltage, and an error voltage of  $\Delta V$  is added to the measured voltage  $V_{sense}$ . With the differential configuration the low line of the measured voltage  $V_{sense}$  is also taken to the input amplifier. This way the error caused by  $\Delta V$  has no influence on the measured signal  $V_{sense}$  when it arrives at the input stage.

Although differential configurations are more complex and expensive to implement, in power system environments they offer better noise immunity than their single-ended counterparts. For this reason extensive use is made of differential configurations in the developed system.

### 2.3.7 First-in, first-out (FIFO) buffers

First-in, first-out (FIFO) buffers are often used in data acquisition system to ensure that no data is overwritten or lost. While an ADC is busy sampling, data bytes are continually generated and need to be stored. If, for some or other reason, the storage process is interrupted for a moment, a few samples from the ADC may be lost. Gap-free and continuous

Table 2-2 Parallel port modes [24]

Parallel Port Mode	Direction of data flow
Compatibility (SPP)	PC to peripheral
Nibble	Peripheral to PC
Byte	Peripheral to PC
Extended Capabilities Port (ECP)	Half duplex
Enhanced Parallel Port (EPP)	Half duplex

Compatibility mode is also known as standard parallel port mode or *Centronics* mode [24]. A software driver places the data on the port's data lines, checks the peripheral for errors and that it is not busy and then clocks the data to the peripheral. In order to output one byte of data it requires at least four I/O instructions, which limits the bandwidth of the port to the order of 150 kbytes per second [25].

Nibble mode is used to transfer data from a peripheral into the parallel port. The 5 status lines (in all standard parallel ports) from the peripheral to the PC are used for sending data [25]; thus a byte (8 bits) must be sent as two nibbles (4 bits) to the PC. As with compatibility mode, a software driver is used to set and read the parallel port lines. Nibble mode is the most software intensive mode for reverse data communications (peripheral to PC). There is therefore a limitation of approximately 50 kbytes per second for this type of data transfer [25].

Byte mode is another port mode used for reverse data transfers (peripheral to PC). During byte mode the drivers used for driving the data lines of the parallel port are disabled and the data port becomes an input port [25]. An entire byte of data can then be sent to the PC in one data transfer cycle, rather than the two cycles as required for nibble mode. Again a software driver has to write the data, check the handshake lines and assert the appropriate control signals. Data rates approaching that of compatibility mode can be obtained by configuring the parallel port for byte mode [25].

High data transfer rates can be obtained with enhanced parallel port (EPP) mode. EPP mode makes use of interlocking handshakes [25], which means that each control signal transition is immediately acknowledged by the opposite side of the interface. An I/O controller handles all the handshaking and data transfers, enabling one data transfer cycle to occur within only one I/O cycle [25]. Consequently, data transfer rates from 500 kbytes per second up to 2 Mbytes per second can be obtained through an EPP port [25].

Extended capabilities port (ECP) mode also uses hardware to assist in data transfers between the PC and peripherals [26]. The major difference between ECP and EPP mode is that direct memory access (DMA) is employed to drive the ECP port rather than explicit I/O operations [26]. ECP is also a loosely coupled interface in the sense that the software driver does not know the exact state of the data transfers. Its only concern is whether the transfer was completed or not [25]. The data rates that can be obtained with ECP mode are similar to those obtainable with EPP mode [25].

A logic analyser, manufactured by *Jobmatch*, is used to download the captured data from the prototype acquisition system to the PC. The *Jobmatch* device makes use of the PC parallel port, configured in standard parallel port (SPP) mode. The utilisation of the logic analyser and the constraints involved with this method of downloading are described in chapter 7. The high data transfer rates of extended capabilities port (ECP) mode make it an attractive option to be considered in the future for integration into the developed system.

#### **2.3.8.3 Direct connection to an internal bus**

Instead of using the standard ports of PCs for interfacing to the acquisition hardware, a dedicated board can be designed to plug in directly onto the ISA or PCI bus of the computer. Depending on the speed of the data bus and the motherboard of the PC, data transfers between microprocessor and memory can then occur at rates in the MHz range (up to 40 MHz). It would be possible to design the system so that one dedicated board controls and downloads data from a network of acquisition devices. However, a major disadvantage of such an approach is that the driver software is very complex and time-consuming to develop. Furthermore, a single error in the software could result in a complete system failure. This option was therefore not considered in the design of the acquisition system prototype. It is, however, a configuration to be considered for the future work in developing a final product.

### **2.4 Topology of the developed system**

Given the considerations discussed in section 2.3, the topology shown in Figure 2-8 was chosen for the developed acquisition system. The system consists firstly of two *isolated probes* that are responsible for the data conversion in the acquisition system. These can be

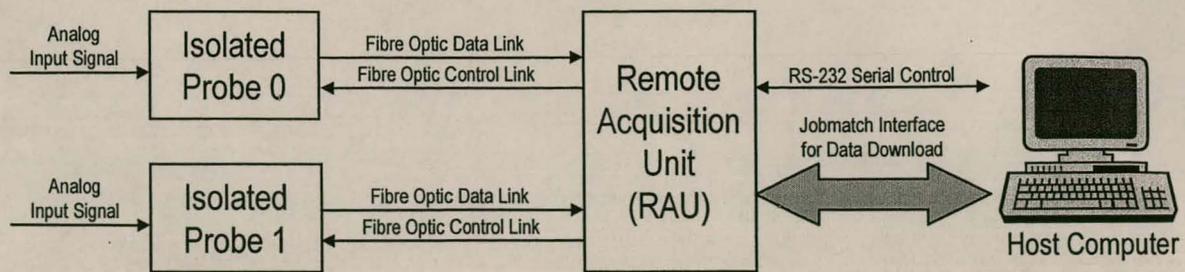


Figure 2-8 System topology for the developed acquisition system

located anywhere in a plant or substation. The probes each samples an analog signal and the data is then sent through optical fibre data links and received centrally at a *remote acquisition unit* (RAU). The RAU contains the storage module of the acquisition system, where the received data from the probes is saved in real time. Once the data has been collected it can be downloaded to the host computer to be displayed and/or analysed.

The RAU is controlled from the host PC via the PC serial port. The RAU in turn controls the *isolated probes* via the fibre optic control links. This configuration ensures that the inputs of the acquisition hardware are fully isolated from each other. The functional description of the various system components is discussed in further detail in the following subsections.

#### 2.4.1 Isolated probes

The functions of the isolated probes can be summed up as follows:

- Digitise analog input signal at the chosen conversion speed.
- Serialise and encode the sampled data in real time.
- Transmit serial data across the optical fibre data link.

An analog-to-digital converter is implemented on each isolated probe for digitising the analog input signal. The specifications of the ADCs are discussed in section 2.2. This data is then serialised for transmission across the fibre optic data link. It is also encoded before transmission in order to improve the integrity of the data at the receiving end of the optical link. Any error that does occur during transmission is detected on the receiving end in the RAU.

#### 2.4.2 Remote acquisition unit (RAU)

The remote acquisition unit (RAU) is the heart of the data acquisition system. It is responsible for the following:

- Providing a data and control interface to the host computer.
- Controlling the operation of the two isolated probe controllers.
- Receiving the optical data from the two probes.
- Storing the sampled data in real time in on-board memory.

The RAU interfaces to the host computer and to the two isolated probes. It is controlled from the computer and in turn controls the operation of the isolated probes. There is both a serial and parallel connection between the host computer and the RAU. The serial connection is used for controlling the RAU, while data is downloaded from the RAU to the PC via the parallel link. Note that the parallel connection has been implemented solely for testing purposes.

The host computer transmits the data acquisition criteria, such as the sampling speed of the isolated probes, the number of active channels, etc. to the RAU. The isolated probes are then activated via the fibre optic control links from the RAU, and the acquisition cycle is executed.

Figure 2-9 gives a high-level block diagram of the RAU.

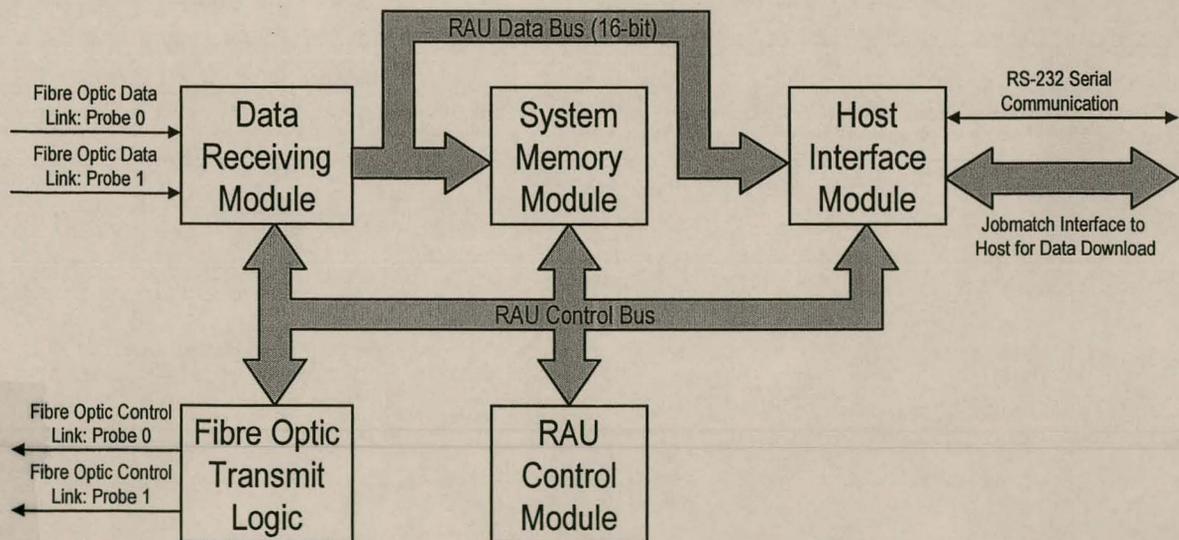


Figure 2-9 High-level block diagram of the remote acquisition unit (RAU)

The isolated probes are controlled from the *RAU control module* via the fibre optic transmit logic. During data acquisition, the sampled data from the two isolated probes arrive at the

*data receiving module* (DRM) where it is decoded and checked for transmission errors. The recovered data is then transferred to the *system memory module* (SMM) where it is stored in real time. The DRM contains a FIFO buffer to ensure that no data bytes are lost between the DRM and SMM. The *RAU control module* also provides the control signals for the DRM and the SMM. Once the data has been captured it is downloaded to the host via the *host interface module*. During configuration of the RAU this module also provides the interface between the PC and the RAU.

### 2.4.3 Host computer

A software program has been developed to run on the host computer for accessing the acquisition hardware. The host computer is responsible for:

- Enabling the user to set the data acquisition criteria.
- Enabling the user to initialise and terminate acquisition and download cycles.
- Configuring the RAU.
- Downloading the collected data from the RAU and saving it to a file.
- Displaying the collected data.

The software environment of the acquisition system allows the user to specify data acquisition criteria such as the sampling speed, the number of active probes, etc. The user also initiates an acquisition cycle from this environment. Once the data capturing process has been executed, the user can download the acquired data to the host via its parallel port.

## Chapter 3

### Isolated probes

#### 3.1 Overview

The data acquisition system is designed specifically for use in high voltage environments. For this reason, aspects such as electromagnetic interference and voltage isolation had to be considered when determining the topology of the system. As discussed in chapter 2, many of these problems can be eliminated by implementing fibre optic links between the digitising probe and the remote acquisition circuitry. The isolated probes refer to the subsystems that digitise and transmit the analog input signals and then transfer the resulting data to the remote acquisition unit (RAU) via fibre optic data links. This chapter discusses implementation of these isolated probes.

Figure 3-1 shows a block diagram of the topology of an isolated probe.

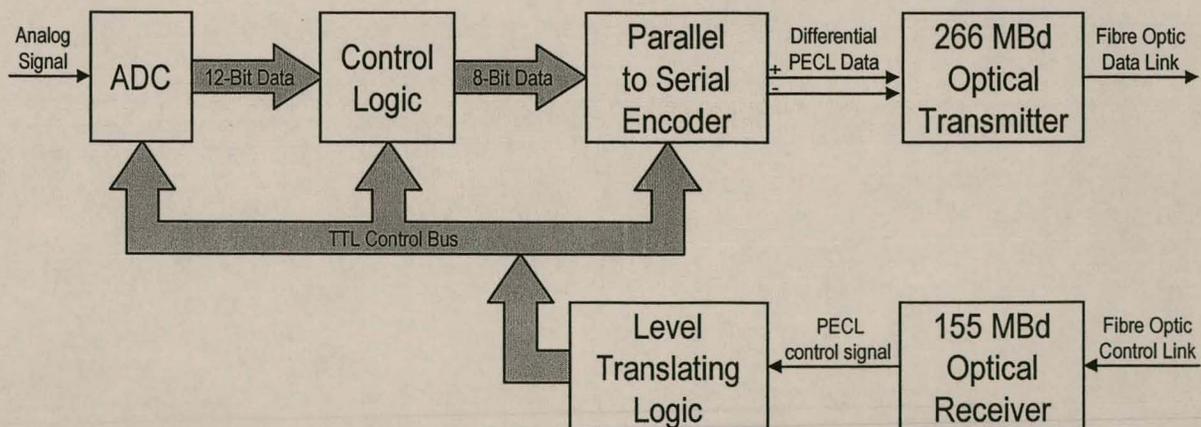


Figure 3-1 Block diagram of the isolated probe

An analog-to-digital converter (ADC) samples and digitises the analog input signal, with a resolution of 12-bits and maximum sample rate of 5 MHz. The control signals for the ADC are generated by the control logic in Figure 3-1. The control logic then transfers the sampled

digital data to a parallel-to-serial encoder in the form of two 8-bit bytes. Each data byte is serialised and encoded to yield a 10-bit word, and then transmitted via a 266 MBd fibre optic link to the remote acquisition unit (RAU).

Each isolated probe is in turn controlled from the RAU via a 155 MBd fibre optic control link. During data acquisition the conversion clock for the ADC is sent across this link. The optical receiver produces a positive emitter-coupled logic (PECL) signal, which is converted back to transistor-transistor logic (TTL) by the level translating logic.

### 3.2 Analog-to-digital converter (ADC)

An ADS801 analog-to-digital converter, manufactured by *Burr-Brown*, is implemented on each isolated probe. The ADS801 is a high-speed sampling ADC with maximum sampling frequency of 25 MHz and resolution of 12 bits. The ADCs are never clocked at a rate higher than 5 MHz due to hardware limitations in the system. These limitations are discussed at a later stage.

Figure 3-2 [13] shows the internal architecture of the ADS801.

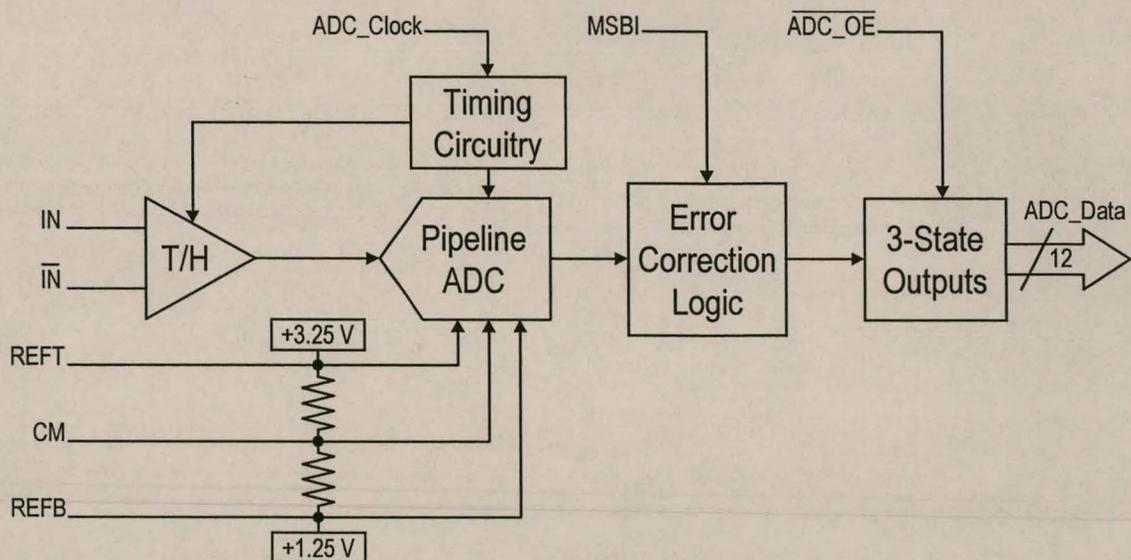


Figure 3-2 ADS801 analog-to-digital converter architecture [13]

ADC\_Clock is the conversion clock for the converter and is used for driving the timing circuitry of the device. The ADS801 accepts a differential input signal with a common-mode

voltage of 2.25 V. The track/hold circuit constructs a DC representation of the differential input signal at the sample time. This signal is quantified in the pipelining ADC and the result is fed to a digital error correction circuit. The data can be transferred to the output pins by driving the *output-enable* (ADC\_OE') input low. When ADC\_OE' is high the 12 data outputs are set to a high impedance state. The *most-significant-bit-inverted* (MSBI) input determines whether the output data is made available in straight offset binary or binary two's complement format.

### 3.2.1 Analog input signal

The differential input of the ADC can be configured in various ways depending on the nature of the analog signal and the performance required. The full-scale range of the input signal is controlled by the internal reference structure shown in Figure 3-3 [13].

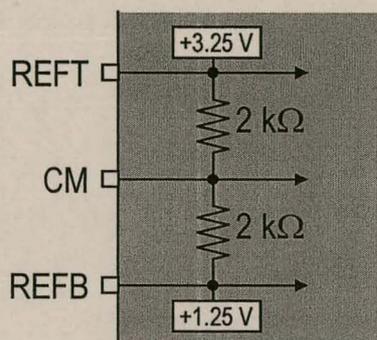


Figure 3-3 Internal reference structure of the ADS801 [13]

The differential inputs are by default centred on a common-mode voltage of 2.25 V, with a full-scale range of 1.25 V to 3.25 V. It is however possible to adjust this range with the *top reference* (REFT) and *bottom reference* (REFB) input pins shown in Figure 3-3. By applying external reference voltages to REFT and REFB, the 1.25 V and 3.25 V references can be overridden. The common-mode voltage will then be set halfway between the two external references. These references are valid as long as REFT is less than or equal to 3.4 V and REFB is greater than or equal to 1.1 V, and the difference between REFT and REFB is greater than or equal to 1.5 V.

The ADS801 can also be configured to receive a single-ended input signal. This is achieved

by connecting the complimentary input (IN') to the common-mode reference voltage (CM). The full-scale range of the analog input signal (IN) is then 0.25 V to 4.25 V. This configuration does, however, cause an increase in even-order harmonics at higher input frequencies. In the developed system a selection can be made with jumper settings between either a differential signal with 2.25 V common-mode voltage, and a single ended input with full-scale range of 0.25 – 4.25 V. The jumper settings are shown in Appendix B.

### 3.2.2 Track/hold circuit

Figure 3-4 [13] shows the configuration of the track/hold circuit in the ADS801.

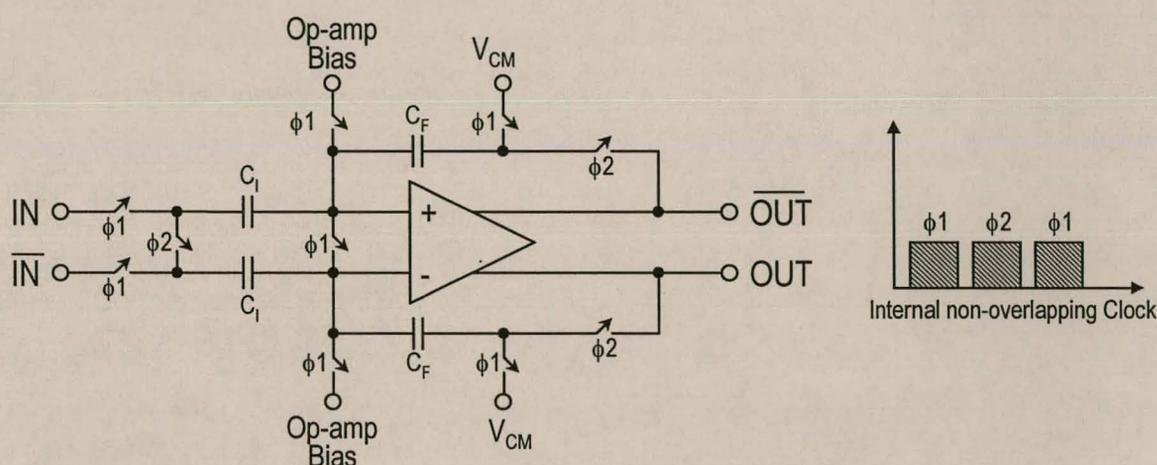


Figure 3-4 Input track and hold circuit [13]

The circuit is controlled with an internal clock, which is a non-overlapping two-phase signal as shown above. During the first phase only the switches marked  $\phi 1$  in Figure 3-4 are closed. The two differential input signals then appear across the input capacitors ( $C_i$ ). At the same time the feedback capacitors ( $C_F$ ) are loaded with the difference between the Op-amp Bias voltage and the common-mode voltage ( $V_{CM}$ ). The non-ideal properties of the op-amp cause this op-amp bias voltage.

During the second phase the  $\phi 1$ -switches open and the  $\phi 2$ -switches are closed. At this time the stored charge in the input capacitors and feedback capacitors is redistributed between the four capacitors. With this configuration the op-amp bias voltage and common mode voltage does not influence the differential output, as the charge in the two feedback capacitors cancel

out. The output is then a fully differential representation of the input signal at the sample time.

### 3.2.3 Pipelining and digital error correction

The ADS801 has a full differential architecture with pipelining and digital error correction logic to guarantee 12-bit resolution. The architecture of the pipelined quantifier is shown in Figure 3-5 [13].

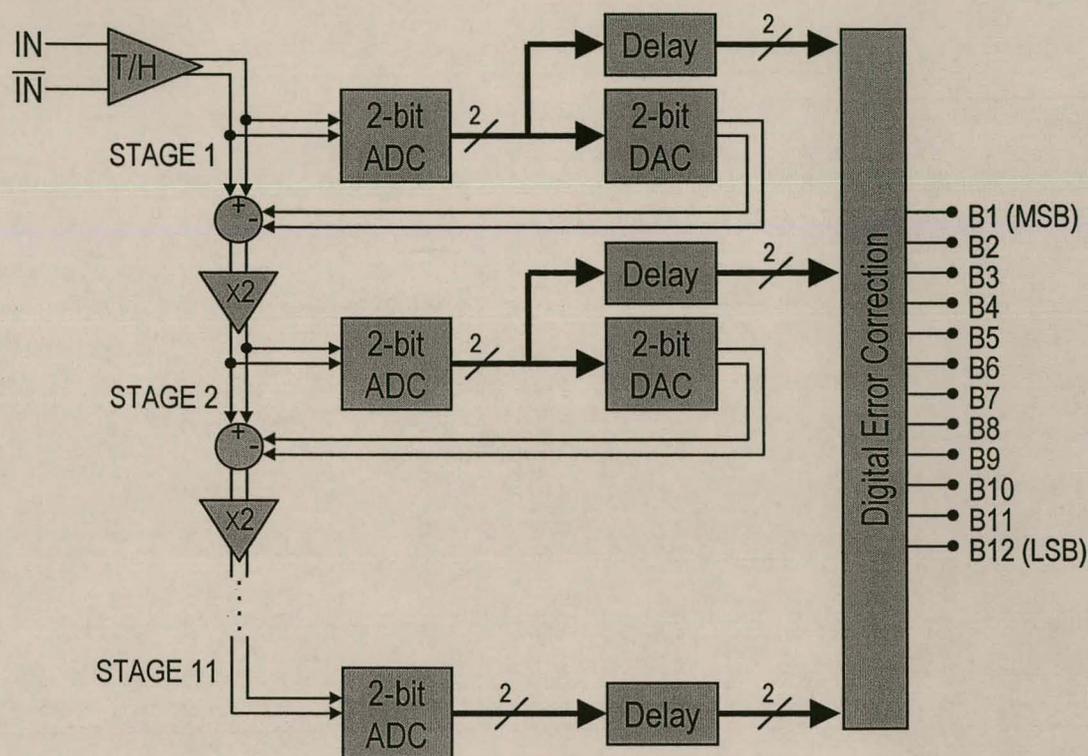


Figure 3-5 Pipeline architecture and error correction logic [13]

The pipeline architecture of the ADC consists of 11 stages each containing a 2-bit flash ADC and a 2-bit digital-to-analog converter (DAC). The 2-bit ADCs convert on the edge of a sub-clock that is twice the frequency of the externally applied clock (ADC\_Clock). The digital output of each ADC is delayed so as to align it with the data created from the next stages of the pipeline. Based on the information found on the redundant bits, the digital error correction logic then adjusts the output data. There is a 6.5 clock cycle data latency from the sample time to a valid data output because of the two pipeline stages per external clock cycle.

The rising and falling edges of ADC\_Clock control the inter-stage conversions in the pipeline. ADC\_Clock must therefore have an accurate 50 % duty cycle with low jitter and rise/fall times of 2 ns or less. In the developed system the clock signal is supplied from the RAU control module and routed through the control logic block. This routing of the clock signal is discussed further in section 3.5.

The conversion technique used in the ADS801 yields the high performance of this ADC. The 2-bit flash converters enable conversion rates in the MHz range, but the cost of the device is minimised by not implementing a full 12-bit flash converter. The pipeline technique gives the ADS801 excellent differential linearity and guarantees no missing codes at 12-bit resolution.

### 3.2.4 Digital output data

The digital output data is provided at CMOS logic levels. The data can be coded as either straight offset binary (SOB) or binary two's complement (BTC), depending on the level of the *most-significant-bit-inverted* (MSBI) pin. Table 3-1 [13] gives the coding table for the ADS801.

Table 3-1 Coding table for the ADS801 [13]

Differential Input	Output Code	
	MSBI = 0 (SOB)	MSBI = 1 (BTC)
+ Full Scale	111111111111	011111111111
+ Full Scale – 1 LSB	111111111111	011111111111
+ Full Scale – 2 LSB	111111111110	011111111110
+ ¾ Full Scale	111000000000	011000000000
+ ½ Full Scale	110000000000	010000000000
+ ¼ Full Scale	101000000000	001000000000
+ 1 LSB	100000000001	000000000001
Bipolar zero	100000000000	000000000000
- 1 LSB	011111111111	111111111111
- ¼ Full Scale	011000000000	111000000000
- ½ Full Scale	010000000000	110000000000
- ¾ Full Scale	001000000000	101000000000
- Full Scale + 1 LSB	000000000001	100000000001
- Full Scale	000000000000	100000000000

If MSBI is low or floating (internal pull-down resistor) the data coding is straight offset binary and a full-scale input corresponds to all '1's at the output. A high on MSBI provides binary two's complement outputs, where the most significant bit is inverted. The digital outputs can

also be set to a high impedance state by driving the *output-enable* (ADC\_OE') pin high. In the developed system MSBI is left floating (internal pull-down resistor), and ADC\_OE' is controlled from the control logic module.

The timing of the output data relative to the conversion clock is shown in Figure 3-6.

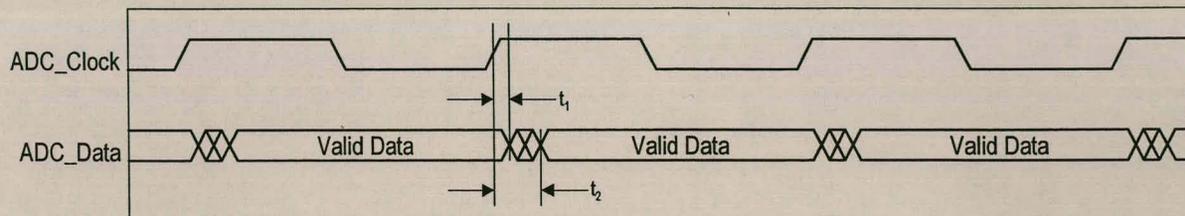


Figure 3-6 Timing specifications for the ADS801 output data

After a rising edge on ADC\_Clock, the data currently at the output of the ADC stays valid for at least a further  $t_1$  nanoseconds. Thereafter the output switches to the next 12-bit word in the ADC pipeline. The new data is guaranteed to be valid  $t_2$  nanoseconds after the rising edge of ADC\_Clock. The minimum value of  $t_1$  is specified as 3.9 ns and the maximum of  $t_2$  as 12.5 ns.

### 3.3 Parallel-to-serial encoder

The ADC provides 12-bits of parallel data at CMOS logic levels. This data must be transmitted across the 266-MBd optical fibre transmitter, which accepts serial positive emitter-coupled logic (PECL) data. A parallel-to-serial encoder is implemented between the ADC and the optical transmitter. The function of the encoder is to serialise the data from the ADC and to provide a PECL interface to the 266 MBd optical fibre transmitter. The above-mentioned objectives can easily be obtained by implementing the CY7B923 HOTLink™ transmitter by *Cypress*. The logic block diagram of this device is shown in Figure 3-7 [10].

Eight bits of data are loaded into the transmitter at a byte rate equal to the frequency of the clock input (CKW). The data is then encoded into ten bits of data and serially shifted out of the three differential PECL ports (OUTA, OUTB and OUTC) at a rate equal to ten times the byte rate. Various other functions are included in the device to facilitate testing and system integration.

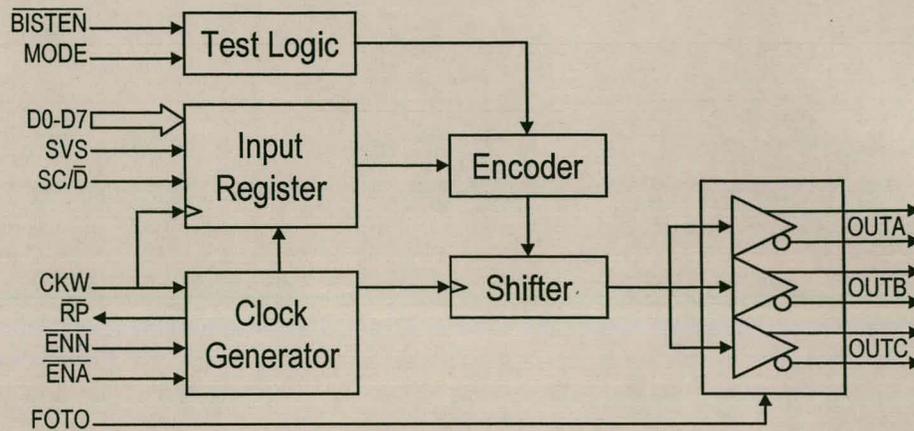


Figure 3-7 CY7B923 transmitter block diagram [10]

### 3.3.1 Input register

The input register holds the data to be processed by the transmitter. On positive edges of CKW the information on the data lines (D0-D7), the *special character/data* line (SC/D') and the *send violation symbol* line (SVS) is loaded into the register. If SC/D' is low, D0-D7 is encoded as a data byte before being sent out at one of the differential serial outputs. A high on SC/D' causes the input data to be transmitted as a control byte. Further details on the data and control bytes are given in section 3.3.3. When SVS is asserted the transmission of a specified violation symbol is forced, allowing for checking of the error handling logic of the HOTLink™ system. This input pin is, however, not used in the developed system and is permanently set at a logical low.

As the CY7B923 transmitter contains only 8 data inputs, the 12-bit data from the ADC needs to be multiplexed in some way before sending it to the transmitter. The control logic module performs this function. For every 12-bit byte from the ADC the 8 most significant bits are transmitted across the link first and then the 4 least significant bits together with 4 zeros. The specific timing of this multiplexing is discussed in section 3.6.2.1. The control logic also supplies the SVS and SC/D' inputs to the CY7B923.

### 3.3.2 Clock generator

The clock generator enables the loading of data into the input register and also supplies the bit

rate clock to the serial shifter. The two enable inputs ENA' and ENN' determine when data is loaded into the input register. If ENA' is asserted, the inputs are loaded into the device on the first rising edge of CKW that follows. On the other hand, if ENN' is asserted, the input data will be loaded into the register on the second rising edge of CKW that follows. If neither ENA' nor ENN' is asserted a special character, namely a *sync-bit*, is transmitted. This character is needed to maintain synchronisation between the HOTLink™ transmitter and receiver and is discussed further in chapter 4.

A 25 MHz crystal oscillator is implemented to provide the clock signal CKW. This enables transmission of the data at a baud rate of 250 MBd. The bit-rate clock for driving the serial shifter is obtained by using CKW and multiplying it by ten. An embedded phase-locked loop (PLL) in the clock generator performs this function.

The *read pulse* (RP') output is only used in systems where D0-D7 is supplied from an asynchronous FIFO. This output is derived from the PLL multiplier and has proper phase and pulse widths to allow correct data transfer between a FIFO and the CY7B923. RP' is not used in the developed system.

The timing for transmission of the ADC data by the CY7B923 is shown in Figure 3-8.

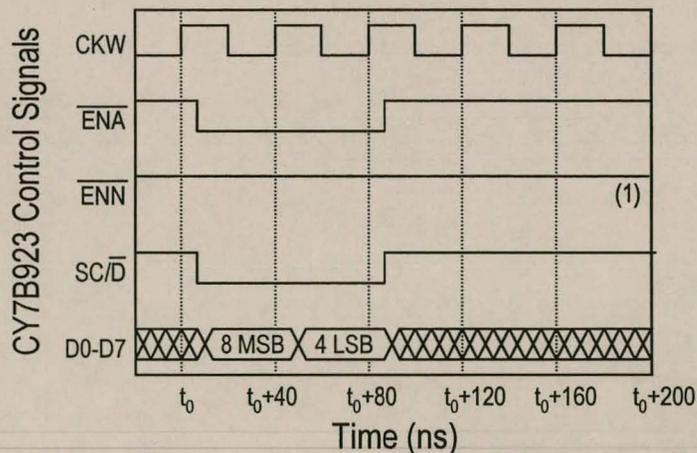


Figure 3-8 CY7B923 control signals for transmitting the ADC data

For each 12-bit sample from the ADC to be transmitted, two 8-bit data bytes are transmitted through the CY7B923. ENA' and SC/D' are therefore pulsed low for a period equal to two clock cycles, thus enabling the transmission of two data bytes. During the first clock cycle

the 8 most significant bits of the ADC data is written to the CY7B923 data inputs. The 4 least significant bits together with 4 zeros are written to the inputs during the second clock cycle. While ENA' is low, two rising edges on CKW are detected at  $t = t_0 + 40$  nanoseconds and  $t = t_0 + 80$  nanoseconds. The two data bytes are transmitted to the optical transmitter on these respective edges.

### 3.3.3 Encoder

The function of the encoder is to transform the data in the input register into a more suitable form for transmission via the fibre optic data link. The coding that is used for transforming the data is given in a Table A-1 in Appendix A. If SC/D' is in a low state, the data on D0-D7 is encoded as 10 bits using the data in Table A-1, and shifted to the differential output ports. Table A-2 in Appendix A shows the encoding for special characters when SC/D' is set high. In such a case the data inputs represent a special control character. In the developed system only the K28.5 special character (*sync-bit*) is used for the synchronisation between the CY7B923 transmitter and the CY7B933 receiver on the RAU. The use of this character is discussed in chapter 4.

It is possible to bypass the 8B/10B coding function of the encoder for systems that include an external encoder. This function is controlled with the MODE input of the transmitter. Encoding of the input data takes place only if the MODE input is low. If MODE is set high, then the 10-bit already encoded data must be supplied to the input register. The data inputs (D0-D7) as well as SC/D' and SVS then become the ten inputs to the shifter. In the developed system MODE is permanently connected to ground.

### 3.3.4 Serial outputs

The three serial interface PECL output buffers, i.e. OUTA $\pm$ , OUTB $\pm$  and OUTC $\pm$ , are the drivers for the optical fibre transmitter. They are all connected to the shifter and contain the same serial data. Only OUTA $\pm$  is used in the developed system. The unused pairs are wired to  $V_{CC}$  to minimise the power dissipated by the output circuit and to minimise unwanted noise generation. Around 5 mA per unused pair is saved with this configuration [15].

$OUTA_{\pm}$  and  $OUTB_{\pm}$  are controllable by the *fibre-optic turn-off* (FOTO) input. If FOTO is asserted then these ports are disabled and their outputs are forced to a logical zero (optical transmitter light switched off). This is convenient for power saving during periods of time when data transmission is inactive.

### 3.3.4.1 Positive emitter-coupled logic (PECL)

Unlike with standard emitter-coupled logic (ECL), positive emitter-coupled logic (PECL) devices operate from a +5 V power supply, allowing the transistor-transistor logic (TTL) and ECL components in a system to operate from a common 5 V supply. Before utilising the PECL outputs of the CY7B923, a full understanding of a PECL output stage is needed. The configuration of a basic PECL output is shown in Figure 3-9 [15].

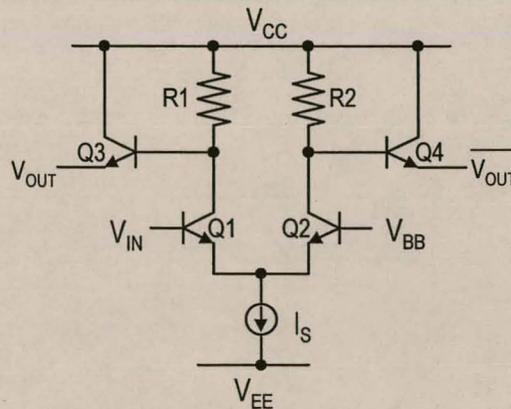


Figure 3-9 Buffered PECL switch [15]

The PECL switch functions with a current source of which the current is directed through either Q1 or Q2. The base of Q2 is internally biased at a fixed voltage  $V_{BB}$ . The level of the input voltage  $V_{IN}$  then determines whether the current flows in R1 or R2. If  $V_{IN}$  is slightly above  $V_{BB}$  (approximately 125 mV), essentially all the current will flow through Q1. If  $V_{IN}$  is 125 mV below  $V_{BB}$ , the majority of the current flows through Q2. An input swing of as little as 250 mV can therefore cause the gate to switch completely from a 0 to a 1. This is why PECL is such a high-speed logic family.

The emitter-follower transistors Q3 and Q4 are included for their very low on-state output impedances (5 – 7  $\Omega$ ) [15]. By implementing these transistor buffers, the PECL gate can drive transmission lines with impedances as low as 50  $\Omega$ . The uncommitted emitters of the

transistors can however not sink any current; therefore it requires a load that operates in the pull-down mode.

### 3.3.4.2 PECL output biasing

The HOTLink™ PECL outputs are specified to drive an equivalent load of  $50\ \Omega$  down to  $V_{CC} - 2\ \text{V}$ . For such a configuration an additional power supply is required. A pair of resistors can also be implemented as shown in Figure 3-10 to create a load that matches that of a single  $50\ \Omega$  resistor connected to  $V_{CC} - 2\ \text{V}$ .

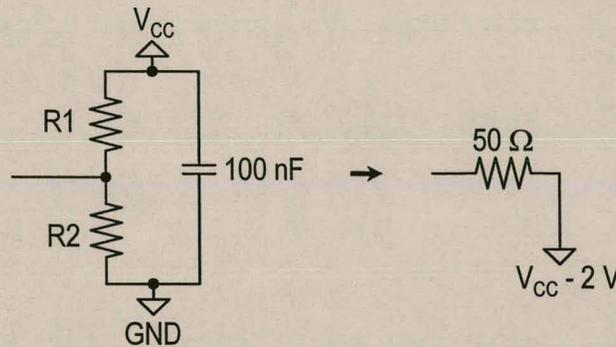


Figure 3-10 PECL output biasing equivalent [15]

The capacitor shown is necessary to facilitate the proper load for AC signals. The values of  $R1$  and  $R2$  are solved using the following equations (see Appendix H):

$$R1 = \frac{V_{CC} \times 50}{V_{CC} - 2} \quad (3.1)$$

$$R2 = \frac{V_{CC} \times 50}{V_{CC} - (V_{CC} - 2)} \quad (3.2)$$

With  $V_{CC}$  equal to  $5\ \text{V}$ ,  $R1$  and  $R2$  are respectively calculated as  $120\ \Omega$  and  $82\ \Omega$ . Appendix B gives the schematic diagram of an isolated probe, which shows the implementation of this biasing arrangement.

## 3.4 Fibre optic data links

The HFBR-1119/-2119 series of high-speed optical devices are used for transmission of the sampled data from the isolated probes to the RAU. On each isolated probe an HFBR-1119T

transmitter module is implemented. The PECL data from OUTA± of the CY7B923 is transferred to the differential data inputs of the HFBR-1119T where it is converted to an optical signal and transmitted across the optical fibre.

The HFBR-1119/2119 modules provide cost-efficient and high-performance serial data communication at a maximum rate of 266 MBd [9]. It is important to take possible conducted electromagnetic interference into consideration when implementing the link, because of the high switching frequencies. A power supply filter circuit is connected to the supply pins of the HFBR-1119T as can be seen in Appendix B. The lines connecting the differential outputs from the CY7B923 to the transmitter module are kept very short and of equal length to prevent signal skew. The link operates with 50 or 62.5 µm core multimode optical fibre over distances of up to 2 km [9].

### **3.5 Fibre optic control links**

With the topology of the data acquisition system, two isolated probes can be connected to a single RAU. In many applications it is desired that the two signals be sampled not only simultaneously, but also synchronously. For this reason, the conversion clocks for the analog-to-digital converters on the isolated probes are supplied from the RAU control module. It is important that isolation is retained between the isolated probes and the RAU. The clock signal is therefore sent via optical fibre control links.

#### **3.5.1 Optical receiver and level translating logic**

An HFBR-1404 optical transmitter and HFBR-2406 optical receiver are used for the fibre optic control links. These devices can transmit/receive data at a maximum data rate of 155 MBd. On the probe boards the HFBR-2406 receives an optical signal and converts it to TTL level to be used as input to the control logic. Figure 3-11 [14] shows the functional blocks implemented for interfacing the HFBR-2406 light-to-voltage converter to digital logic.

The receiver has a low-level analog output related to the incoming optical power by a 7 mV/µW conversion gain [14]. It therefore needs additional external gain stages to increase the amplitude of its output before it can interface to standard logic elements [14]. The logic compatible comparator supplies the interface to the TTL components that follow.

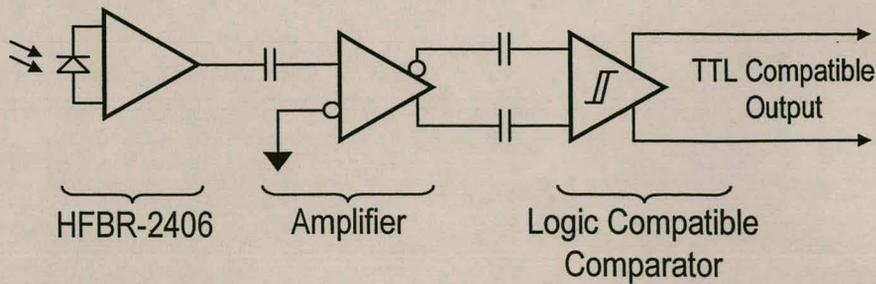


Figure 3-11 Optical fibre receiver interface [14]

The schematic diagrams for the isolated probes, given in Appendix B, illustrate the implementation of the 155-MBd optical link. The typical data rate and achievable distances are given in Figure 3-12.

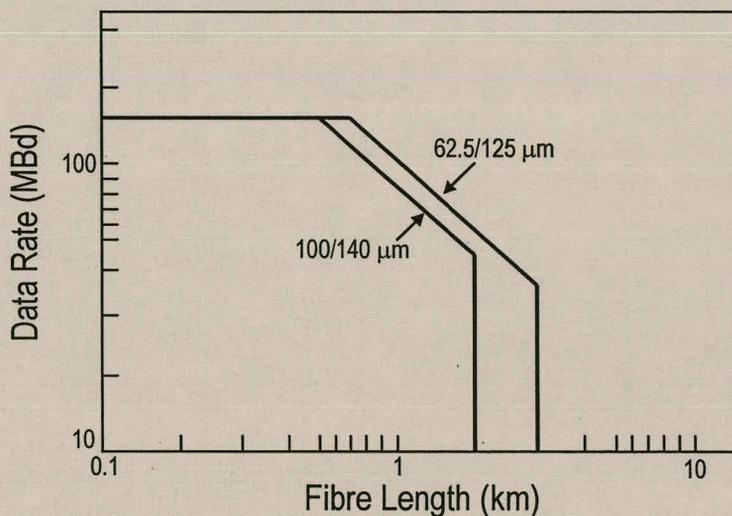


Figure 3-12 Data rate vs. fibre distance for the HFBR-1404/2406 [14]

The link operates with 62.5/125 μm or 100/140 μm optical fibre cable, achieving slightly better response with the smaller core diameter fibre. At full-speed (155 MBd) the length of the optical fibre can be a few hundred metres, and with slower data rates up to 3 km long.

### 3.6 Control logic

The control logic is implemented using a field-programmable gate array (FPGA) manufactured by *Altera*. It controls the operation of all the components on the isolated probe board. The control logic module is in turn controlled from the RAU control module by means of the fibre optic control link. A block diagram representation of the control logic module is

shown in Figure 3-13.

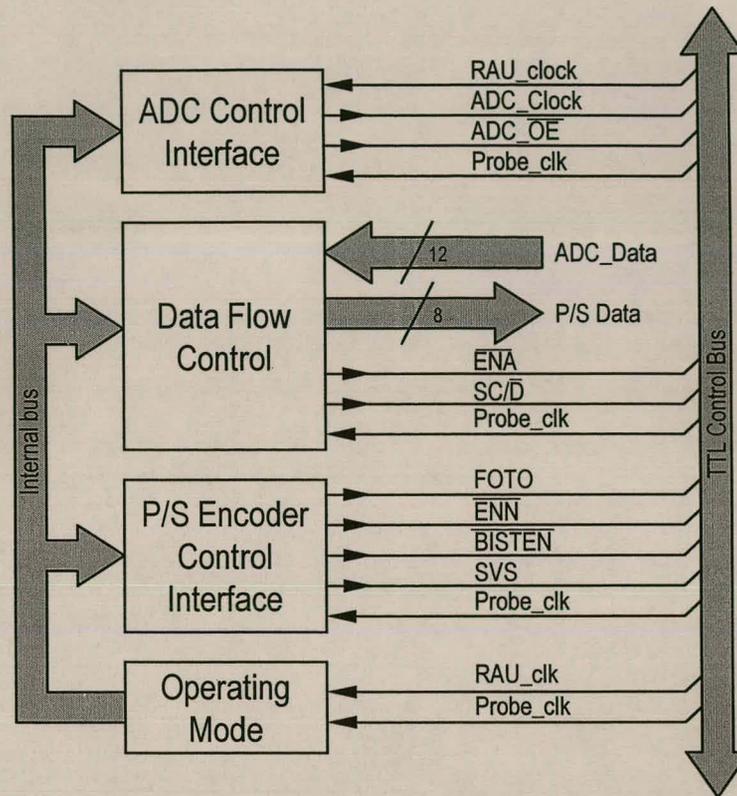


Figure 3-13 Block diagram of the control logic module on an isolated probe

The control logic on an isolated probe operates with a 25 MHz master clock (Probe\_clk) that is supplied from an on-board crystal oscillator chip. The operating mode of the isolated probe is determined by the fibre optic control signal from the RAU (RAU\_clk). This signal is sampled on every positive edge of Probe\_clk to determine whether a sampling clock is being sent from the RAU or not. The control and data lines to the ADC and the parallel-to-serial encoder are configured as determined by the operating mode. The following section discussed the various operating modes of the probes.

### 3.6.1 Operating modes

The isolated probe can be functioning in one of two possible operating modes:

- Acquisition mode.
- Idle mode.

The choice of these modes is determined by the topology of the data acquisition system and

specifically of the isolated probe. If a sample clock is received from the RAU across the fibre optic control links (RAU\_clk), the isolated probes switch to *acquisition mode*. The ADCs are then clocked at the sample clock rate and the digital data is transmitted back to the RAU via the fibre optic data links.

Once an acquisition cycle is complete the RAU returns RAU\_clk to a logical low. The allowable sampling frequency of the ADCs is between 10 kHz and 5 MHz. At a frequency of 10 kHz the period of the signal is 100  $\mu$ s. If RAU\_clk is zero for a period longer than 50  $\mu$ s, the RAU is no longer transmitting the sample clock signal across the control link. The isolated probes then switch to *idle mode*. If RAU\_clk, however, switches high before 50  $\mu$ s has passed a sampling clock is being sent from the RAU. The operating mode is then set to *acquisition mode* and RAU\_clk is gated through to the ADCs as ADC\_Clock.

### 3.6.2 Executing operating commands

The control logic module must supply the control signals to both the ADC and the parallel-to-serial encoder as determined by the operating mode of the probe. ADC\_Clock and ADC\_OE' are supplied from the control logic to the ADC. The 12-bit data bytes from the ADC are transferred to the control logic during *acquisition mode* where it is multiplexed as two 8-bit bytes to the parallel-to-serial encoder. The control signals needed to activate the various operating modes of the CY7B923 are also supplied from the control logic module. The same 25 MHz clock that is used as input for the control logic (Probe\_clk) is used as the CKW input of the CY7B923.

#### 3.6.2.1 Acquisition mode

During *acquisition mode* the conversion clock for the ADC is sent via the 155 MBd link from the RAU. It is routed directly through the control logic to the ADC as ADC\_Clock. The output data lines are enabled by driving ADC\_OE' low. The 12-bit data is transferred to the control logic module and multiplexed as shown in the timing diagram in Figure 3-14.

Here the ADC is sampling at the maximum allowable speed of 5 MHz ( $t_{CONV} = 200$  nanoseconds). As discussed in section 3.2.4 a valid data byte appears at the ADC output

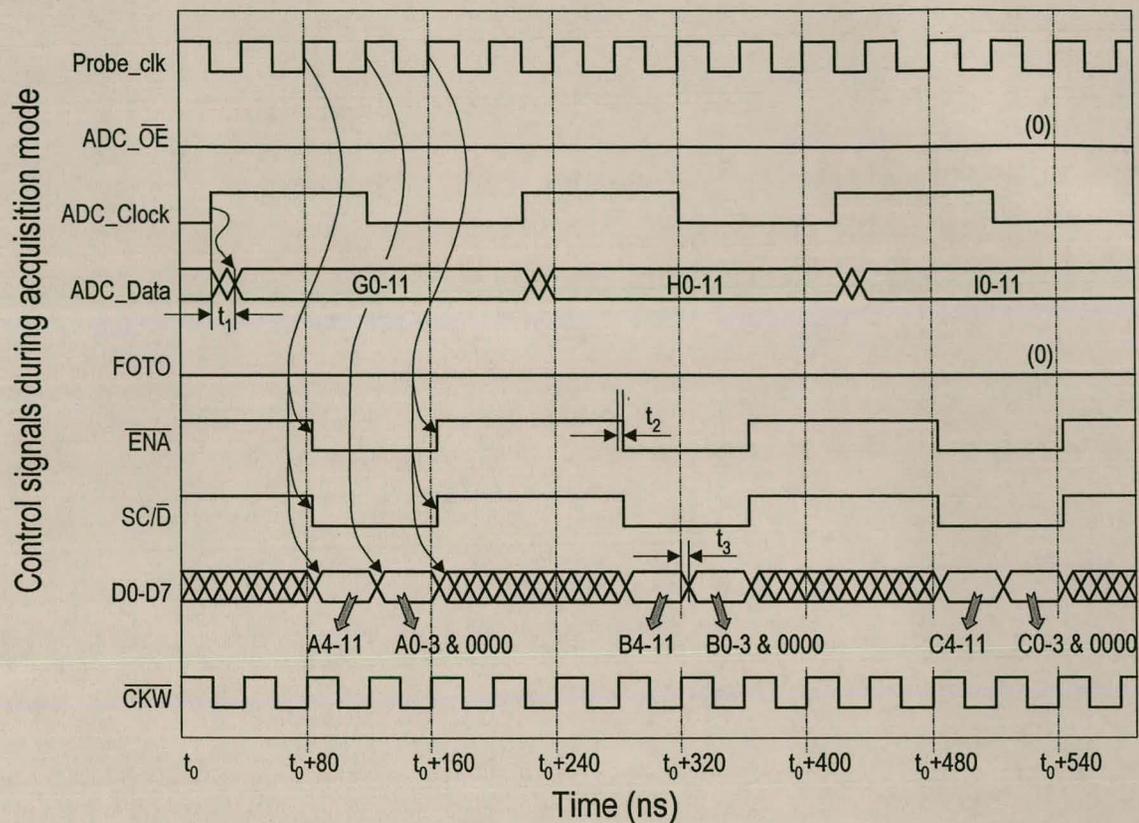


Figure 3-14 Isolated probe timing diagrams during acquisition mode

(ADC\_Data) 12.5 nanoseconds ( $t_1$ ) after every rising edge of ADC\_Clock. This byte on ADC\_Data was sampled at the ADC input 7 conversion cycles prior, due to the data latency of the ADC. The 12-bit byte must then be transmitted across the optical link, via the CY7B923, before the next 12-bit byte appears at the output of the ADC.

The ENA', SC/D' and CKW inputs of the CY7B923 are used for transmission of the data. As discussed in section 3.3.2, if ENA' and SC/D' are low on the rising edge of CKW, the data on the eight data inputs (D0-D7) of the CY7B923 is transmitted to the optical transmitter as a 10-bit serial data stream. It was also mentioned that the 12-bit data from the ADC is transmitted across the link as two 8-bit bytes. The 8 most significant bytes are first transmitted, and thereafter the 4 least significant bits with 4 zeros. The same master clock signal that is used as input to the control logic (Probe\_clk) is used as the CKW input. This ensures that synchronisation is maintained between the control logic and the CY7B923.

The transmission of a single ADC sample can be explained as follows. In Figure 3-14 the first rising edge on ADC\_Clock is detected at  $t=t_0+20$  nanoseconds. Twelve and a half

nanoseconds later ( $t_1$ ) the data on ADC\_Data is a valid data sample (A0-11). On the second rising edge of Probe\_clk after ADC\_Clock is high (at  $t=t_0+80$  nanoseconds), ADC\_Data is latched into the control logic module. As the period of Probe\_clk is 40 nanoseconds, the data on ADC\_Data will certainly be valid by the time that it is latched into the control logic module.

The positive edge of Probe\_clk at  $t=t_0+80$  nanoseconds causes ENA' and SC/D' to pulse low. It also causes A4-11 to be output at the CY7B923 data lines D0-D7 at  $t=t_0+80+t_2$  nanoseconds. The delay  $t_2$  is caused by propagation delays in the control logic. At  $t=t_0+120$  nanoseconds A4-A11 is still present at D0-D7, ENA and SC/D' are low and CKW switches from low to high. This positive edge on CKW causes the encoder to transmit A4-11 to the fibre optic transmitter as a 10-bit serial data byte.

On the rising edge of Probe\_clk at  $t=t_0+120$  nanoseconds, A0-3 together with four zeros are transferred to D0-D7. The data effectively appears  $t_3$  nanoseconds later, as a result of propagation delays in the control logic. On the first rising edge of CKW thereafter, which is at  $t=t_0+160$  nanoseconds, A0-3&"0000" is transmitted. The four bit positions that are filled with zeros are reserved bits. These bits could later be used to contain channel information for example. The transmission of sample A0-11 is ended when ENA' and SC/D' are switched high again at  $t=t_0+160+t_3$  nanoseconds.

### 3.6.2.2 Idle mode

During *idle mode* the ADC on each isolated probe is disabled. The control logic module drives ADC\_Clock low, disabling conversion. ADC\_OE' is driven high to put the ADC outputs in a high impedance state. The inputs to the parallel-to-serial encoder i.e. SC/D', ENN' and ENA' are switched to their non-active states. FOTO is, however, asserted to prevent the transmission of any data across the fibre optic data links.

## Chapter 4

### Data receiver module

#### 4.1 Overview

The data receiver module (DRM) is a subsystem of the RAU as shown in Figure 2-9. From each isolated probe the DRM receives data serially at a baud-rate of 250 MBd. This data is transferred to the SMM, which is also a subsystem of the RAU, and stored in real-time in memory. Two banks of dynamic random access memory (DRAM) packaged as single in-line memory modules (SIMMs), are used as the data storage medium. The SMM is discussed in chapter 5.

The function of the DRM is therefore to receive the incoming serial data from the isolated probes and to convert it to parallel data with TTL logic levels, ready to be transferred to the SMM. Figure 4-1 gives a block diagram of the DRM.

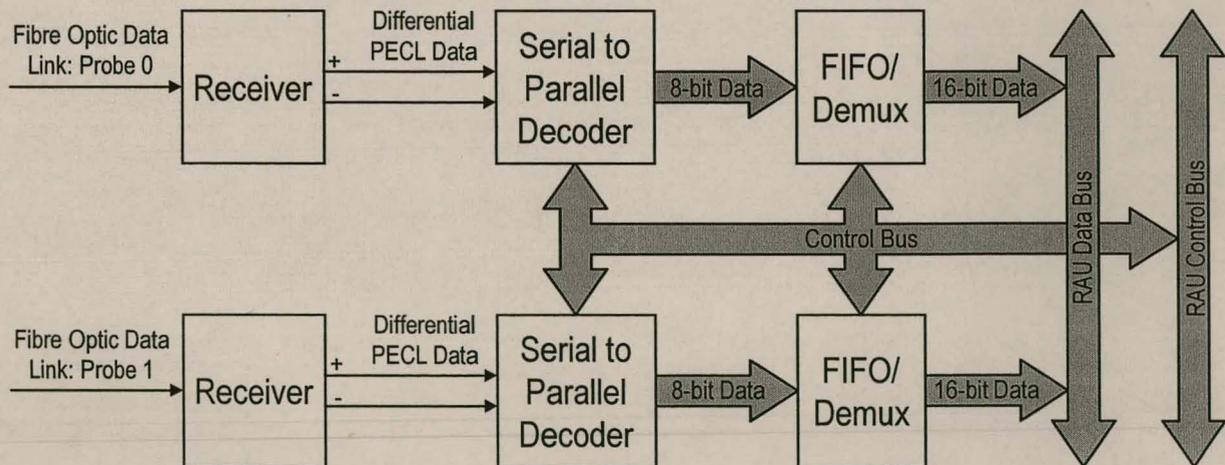


Figure 4-1 Block diagram of the data receiver module (DRM)

The DRM facilitates the reception and conversion of data from two isolated probe channels. The optical signal from each probe is converted to differential PECL logic by means of an optical fibre receiver. A serial to parallel decoder decodes the differential PECL signals and

reconstructs the 8-bit TTL data bytes that were sent from the isolated probes. Two demultiplexers acting as first-in first-out buffers (FIFOs), output the received data bytes on the RAU data bus. Both the serial to parallel decoder and the FIFOs/demuxes are controlled from the RAU control module via the RAU control bus.

The DRAM in the SMM has a 32-bit data interface. It would therefore be ideal if the RAU data bus were 32 bits wide. A 32-bit bus, however, requires complex printed circuit board (PCB) layout and routing. For this reason a 16-bit data bus is rather implemented to simplify board-layout of the RAU. The data on the 16-bit bus is then demultiplexed in the SMM, which is a separate dedicated board, to provide the required 32-bit interface to the DRAM.

## 4.2 Optical fibre receivers

It was mentioned in chapter 3 that the HFBR-1119T optical transmitters and HFBR-2119T optical receivers, connected with 50 or 62.5  $\mu\text{m}$  core multimode fibre, are used for the fibre optic data link. The HFBR-2119T receiver modules are implemented in the DRM. Their function is to convert the optical signals back to differential PECL logic signals as required by the serial to parallel decoders.

The schematic diagram and the printed circuit board-layout for the RAU are given in Appendix C. These schematics show the implementation of the HFBR2119Ts in the system. Note that a low pass filter is connected to the power pins in order to isolate the receiver from noisy dc power.

The HFBR2119T has two pairs of differential outputs, designated as the *signal-detect* outputs and the *data* outputs respectively. The *signal-detect* outputs are not used in the data acquisition system, but the *data* outputs are routed to the inputs of the serial to parallel decoder. The recommended output load for these PECL signals are specified as 50  $\Omega$ . The *data* outputs can therefore be biased in the same way as the CY7B923-transmitter outputs as discussed in chapter 3. The two differential lines are each connected through an 82  $\Omega$  resistor to  $V_{CC}$  and through a 130  $\Omega$  resistor to ground, with a 100 nF capacitor between  $V_{CC}$  and ground.

### 4.3 Serial to parallel decoders

The serial to parallel decoders convert the 250 MBd serial data from the fibre optic link into 8-bit parallel data bytes. The incoming data has been encoded by the CY7B923-transmitters on the isolated probes. Therefore the matching CY7B933 HOTLink™ receivers are used to recover the data bytes. The CYB933s automatically recover the timing information from the serial bit stream, decode the data and check it for transmission errors. Figure 4-2 [10] shows a block diagram of the CY7B933.

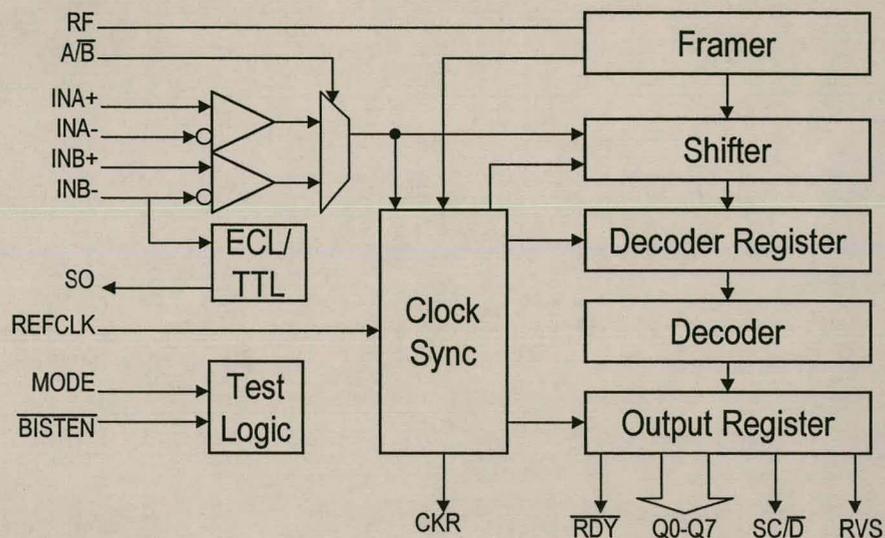


Figure 4-2 Block diagram of the CY7B933 receiver logic [10]

The receiver has two differential data inputs  $INA_{\pm}$  and  $INB_{\pm}$  of which either can be used to receive incoming serial bit streams. A reference clock signal (REFCLK) that is within 0.1 % of the transmitter CKW frequency is supplied to the decoder. This clock signal indicates the rate at which the incoming serial data arrives at the decoder, but does not have to be synchronised with the incoming data. The device then reconstructs the 8-bit data bytes and transfers them to the Q0-Q7 outputs. The various components of the serial to parallel decoder are discussed in the following subsections.

#### 4.3.1 Serial inputs

The serial inputs of the CY7B933 operate at PECL logic levels, making it directly compatible with the outputs of the HFBR-2119T optical receiver. Thus the ECL/TTL converting function shown (SO output) in Figure 4-2 is not needed. Only one of the two line receivers,

namely  $INB_{\pm}$ , is used to receive the incoming serial data. The selection of  $INB_{\pm}$  as the active input is made by setting the  $A/B'$  input of the CY7B923 to a logical low state. This input also operates at PECL logic levels; therefore  $A/B'$  must be set at a PECL low for  $INB_{\pm}$  to be selected as the active input. This is accomplished with a resistive network shown in Figure 4-3 [15].

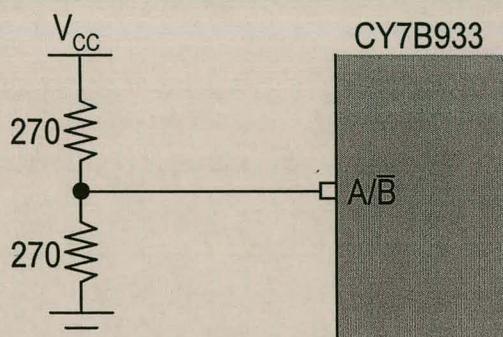


Figure 4-3 Selection of Port B as active differential port for the CY7B933 [15]

With the configuration shown above, the voltage at the  $A/B'$  input is at a PECL logical low (between 2 and 3.525 V) and the  $INB_{\pm}$  differential pair is selected as the active input for receiving the incoming serial data.

#### 4.3.2 Clock synchronisation

For the receiver to interpret the incoming serial data correctly, the serial bits need to be distinguished from each other, and the byte boundaries need to be detected. The framer and the clock synchroniser perform these functions. The reference clock signal REFCLK, with frequency within  $\pm 0.1\%$  of the frequency of the clock that drives the transmitter CKW pin serves as input to the clock synchroniser. An embedded phase-locked loop (PLL) then multiplies REFCLK by 10, providing an internal bit-rate clock. The PLL tracks the frequency of the incoming bit stream and aligns its internal bit-rate clock to the serial data transitions.

The byte boundaries are detected with a *sync-bit* that is periodically sent from the CY7B923-transmitter. The framer in the CY7B933 recognises this special character. It then resets the clock synchroniser to its initial state, causing the data to be framed on the correct byte boundaries. The CKR output of the clock synchroniser is a byte rate clock, which is aligned to the byte boundaries.

The *Reframe Enable* (RF) input must be connected to  $V_{CC}$  to enable *sync-bit* framing. If RF remains high for greater than 2048 bytes, the framer switches to double-byte framing. In such a case the framer requires two *sync-bits* within every five bytes before the synchroniser is reset to its initial state.

### 4.3.3 Decoding of serial data

In Figure 4-2 the shifter is clocked by the clock synchroniser, and receives a bit at a time from one of the differential input pairs. When a complete byte has been received it is sent to the decoder register, which holds the byte until it is transferred to the output register. If MODE is wired to ground, each 10-bit byte is transformed back to its original 8-bit byte in the decoder. The decoder is bypassed when MODE is wired to  $V_{CC}$ .

The switching waveforms for the output register are shown in Figure 4-4.

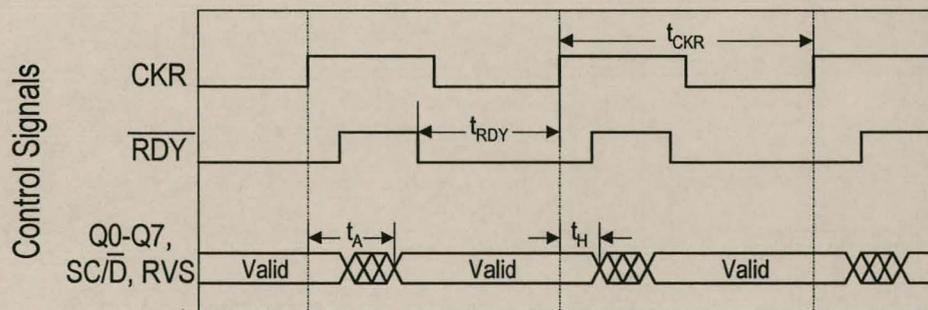


Figure 4-4 Switching waveforms for the CY7B933 output register [10]

The CKR clock output, with period  $t_{CKR}$ , indicates the rate at which data is decoded and output on the Q0-Q7 output pins. The received bytes are first checked for transmission errors. At  $t_A$  nanoseconds after a positive edge on CKR, the *Special Character/Data* (SC/D'), *Read Violation Symbol* (RVS) and Q0-Q7 signals are assigned values. The values are then valid until  $t_H$  nanoseconds after the next rising edge of CKR, as shown above.

If a data byte has been received the SC/D' signal is assigned '0', otherwise it is assigned '1'. The RVS output is set high if an invalid data byte has been received. The output register holds the recovered data (Q0-7, SC/D' and RVS) and then aligns it with the recovered byte clock CKR. Once the recovered data is available at the output, *Data Output Ready* (RDY') is pulsed low for

a period of at least  $t_{RDY}$  before the next rising edge of CKR. The maximum/minimum specifications of  $t_{CKR}$ ,  $t_{RDY}$ ,  $t_A$  and  $t_H$  are given in the data sheets of the CY7B933.

#### 4.3.4 Implementation

The schematic diagram of the RAU in Appendix C illustrates the implementation of the CY7B933-receivers in the data acquisition system. The differential pair  $INB_{\pm}$  is used to receive the serial data from the optical fibre receiver. The  $INA+$  input is terminated to  $V_{CC}$  through a 1.5 k $\Omega$  resistor to prevent stray noise on the other differential pair to be interpreted as serial data [27]. The 8-bit data outputs and the CKR, SC/D', RVS and RDY' control signals from the CY7B933 are connected directly to the FIFO/demux. The FIFO/demux then utilises the control signals to transfer the data bytes to the 16-bit data bus.

#### 4.4 FIFOs/demultiplexers

The function of the FIFOs/demuxes is firstly to latch in two 8-bit data bytes from the CY7B933s and to output them to the RAU data bus as 16-bit words. Such a 16-bit word contains a 12-bit data sample and four zeros, as was sent from the isolated probes. The four zeros could later be replaced with status bits, e.g. channel numbers or gain value information. If both channels are active, a word from probe 0 is written onto the RAU bus and then a word from probe 1, etc. The data can then be saved in the SMM in real-time.

Secondly, the FIFOs/demuxes operate as first-in first-out buffers (FIFOs). This is needed because the data from the two isolated probes arrive at DRM at the same speed, but the data sets are not fully synchronised. A small buffer is therefore needed to ensure that no bytes are lost when the system is running at full speed.

The FIFOs/demuxes are implemented in erasable programmable logic devices (EPLD) with *Very High Speed Integrated Circuit (VHSIC) hardware description language (VHDL)* code. Figure 4-5 gives a block diagram of the FIFO/demux that receives data from probe 0. The FIFO/demux for probe 1 is identical to this one except that the control signal names begin with P1 instead of P0.

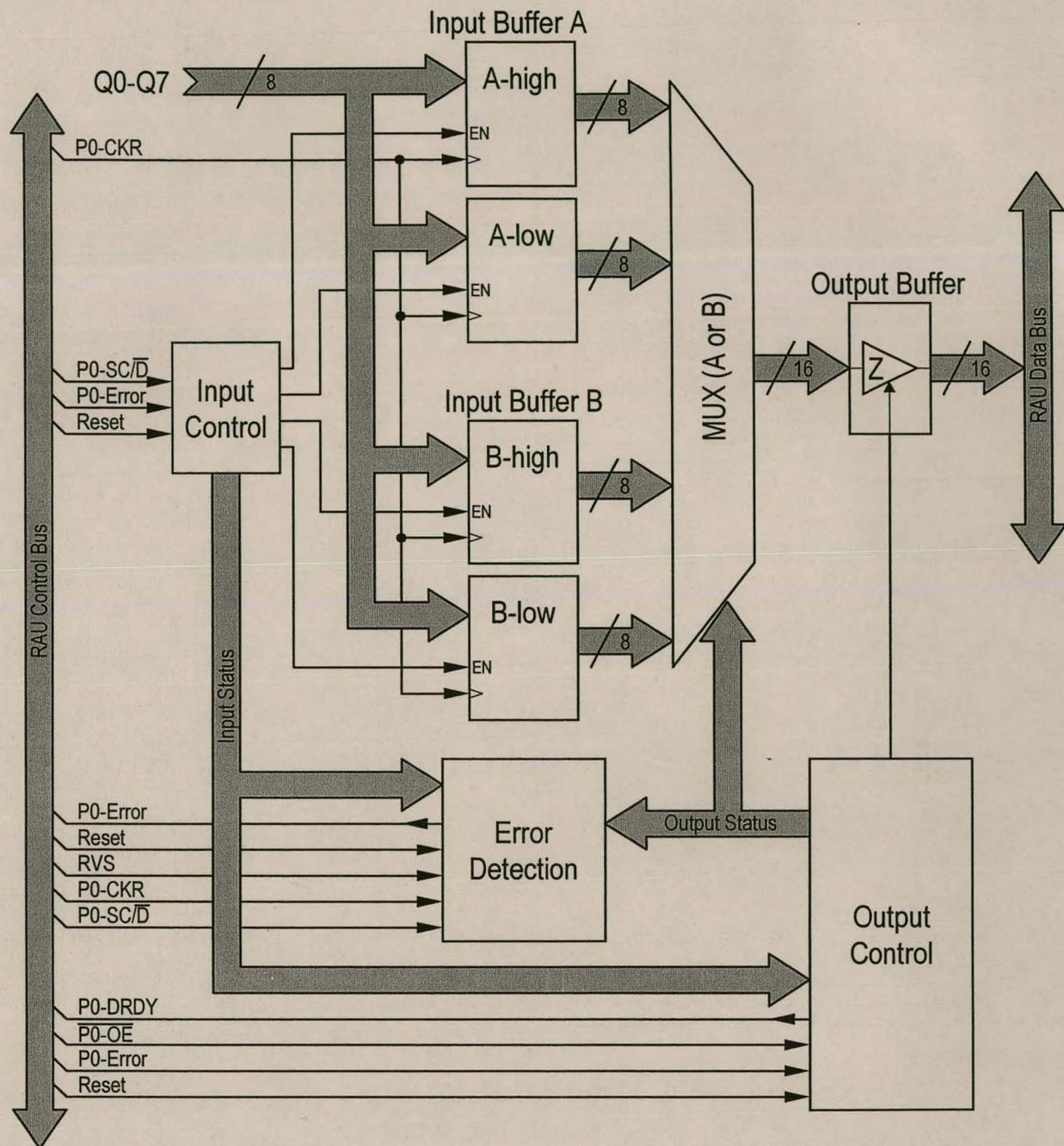


Figure 4-5 Block diagram of a FIFO/demultiplexer

the positive edges of P0\_CKR. A data byte present on the data lines on a rising edge of P0\_CKR (P0\_SC/D' is low) is latched into the upper or lower half of either *input buffer A* or *input buffer B*. The *input control* logic (see section 4.4.1) determines the destination of the data, i.e. A-high, A-low, B-high or B-low. When two bytes have been latched into one of the *input buffers*, the 16-bit byte in that buffer is transferred to the *output buffer*. The *output control* logic respectively controls and monitors *data ready* (P0\_DRDY) and *output enable* (P0\_OE'). Section 4.4.2 discusses the operation of the *output control* logic.

The *error detection* unit in Figure 4-5 is included to detect if any of the received data is corrupted or if data bytes are incorrectly overwritten inside the FIFO/demux. It monitors the *input* and *output control* logic as well as the *read violation symbol* (P0\_RVS), P0\_CKR and P0\_SC/D' signals from the CY7B933. The various error situations are discussed in section 4.4.3. Once an error is detected, the P0\_Error output is asserted and operation of the FIFOs/demuxes is disabled. Reset then needs to be asserted before operation can continue again.

#### 4.4.1 Input control

The *input control* logic is responsible for the flow of data from the CY7B933 into the FIFO/demux. A valid data byte from the CY7B933 is indicated when P0\_SC/D' is low on the rising edge of P0\_CKR. In such a case the data byte is latched into the FIFO/demux and stored in either *input buffer A* or *input buffer B*.

The *input control* logic contains a 2-bit counter, namely *input\_counter*, that determines whether the data is stored in A-high, A-low, B-high or B-low. Initially *input\_counter* is reset to zero. Thereafter it is incremented each time a byte is latched into the FIFO/demux. If the counter is equal to 0, the byte will be stored in A-high. If it is 1, A-low will be used. Similarly, counter values of 2 and 3 cause the data to be stored in B-high and B-low respectively. When *input\_counter* reaches the value of 3 it wraps around to zero again.

#### 4.4.2 Output control

The output of data from the FIFO/demux is also controlled with a counter, namely the *output\_counter*. The value of this counter determines whether the data in *input buffer A* or *input buffer B* is transferred to the *output buffer*. Initially *output\_counter* is reset to 0. The data in *input buffer A* is reproduced in the *output buffer*. As soon as this data is transferred to the RAU data bus (P0\_OE' pulses low and high again), the *output\_counter* is toggled to 1 and the data in *input buffer B* appears in the *output buffer*.

The transfer of data from the FIFO/demux to the data bus is controlled with the *data ready* (P0\_DRDY) and *output enable* (P0\_OE') lines in Figure 4-5. P0\_DRDY signals to the controller when a valid data byte is present in the *output buffer*. The RAU control module then asserts

P0\_OE' via the RAU control bus and the data in the *output buffer* is transferred to the RAU data bus.

The *input\_counter* and *output\_counter* are used to calculate when a data byte in the output buffer has not yet been written transferred onto the data bus. Initially both counters are reset to zero. As long as the *input\_counter* is smaller than 2, *input buffer A* is not yet filled with two valid data bytes. As soon as it switches to 2, the data in *input buffer A* is transferred to the *output buffer* to be written onto the data bus. The *output control* logic pulses P0\_DRDY high, which effectively requests a P0\_OE' pulse from the RAU control module. The control module reacts by pulsing P0\_OE' low for a sufficient time period. When P0\_OE' returns high again the *output\_counter* is toggled to 1 and P0\_DRDY is returned to zero.

In the mean time *input buffer B* has been filling up with data bytes. If a data byte has been latched into B-high, the *input\_counter* has a value of 3. When a second byte is latched into B-low the *input\_counter* wraps around to zero and data must again be written out of the FIFO/demux. The *output control* logic continually compares the values of the *input\_counter* and the *output\_counter*, and detects at this time that P0\_DRDY has to be pulsed high. A '1' on P0\_DRDY in turn causes the RAU control module to pulse P0\_OE' low. As P0\_OE' returns high again *output\_counter* toggles to zero, P0\_DRDY returns low and the cycle starts once again.

#### 4.4.3 Error situations

There are three situations that could arise during an acquisition process where an error occurs and should be indicated to the control module. Firstly, if a data byte has been corrupted during transmission from the isolated probe it will be indicated on the *Read Violation Symbol* (P0\_RVS) line from the CY7B933.

Secondly, if data arrives at the FIFO/demux at a higher rate than at which it can be transferred to the RAU data bus, data in the FIFO/demux will be incorrectly overwritten. Such an error occurs if new data has to be written into one of the two *input buffers*, and the data currently in that buffer has not yet been transferred onto the data bus. The *input\_counter* and *output\_counter* are again monitored for this type of error. If the *input\_counter* is incremented from 3 to 0 it means that the next data byte is going to be stored in *input buffer A*. If at this

time the *output\_counter* is still 0, the previous data byte in *input buffer A* has not yet been written onto the RAU bus and an error has occurred. Similarly, if the *input\_counter* is incremented from 1 to 2 and the *output\_counter* is still equal to 1, the previous byte in *input buffer B* has not yet been output and an error has occurred.

In chapter 3 it was mentioned that the 12-bit data samples from the isolated probes are sent to the RAU as two 8-bit bytes. The first byte contains the eight most significant bits of the sample and the second contains the four least significant bits together with four zeros. The FIFO/demux in the DRM rejoins the two 8-bit bytes to form the original 12-bit data sample plus four zeros. It is essential that the correct data bytes are connected together, and that a byte from one sample is not joined with a byte from another sample.

The input control logic monitors P0\_SC/D' to ensure that the correct bytes are connected together and that the original data samples are reconstructed. On the isolated probes, the two 8-bit bytes containing information on a specific data sample are transmitted on two consecutive positive edges of P0\_CKW (see Figure 3-8). In the time until the next two bytes are transmitted, the *sync-bit* special character is transmitted and P0\_SC/D' is high. In the DRM the two data bytes should therefore be received on two consecutive clock edges of P0\_CKR. If P0\_SC/D' is low on only one edge of P0\_CKR at a time, an error has therefore occurred.

In any of these cases P0\_Error is pulsed high, indicating to the RAU control module that an error has occurred. The isolated probes are then deactivated, thus stopping the acquisition process completely. The FIFO/demux is also deactivated until an external Reset is received from the control module.

#### 4.4.4 Implementation

The FIFOs/demuxes are implemented in EPM7064SLC44-7 EPLDs from *Altera*. Appendix C gives a schematic diagram of the RAU and shows the pin-assignments for these devices. The VHDL-code is given in Appendix D whereas Appendix E gives some information on the different methods of programming *Altera* EPLDs.

## Chapter 5

### System memory module (SMM)

#### 5.1 Overview

The sampled data from the two isolated probes is saved in the system memory module (SMM) during an acquisition cycle. The SMM receives the data from the data receiving module (DRM) and saves it in real-time in dynamic random access memory (DRAM). Once the acquisition cycle is complete, the data is downloaded from the SMM to the host computer via the host interface module. The SMM is controlled from the RAU control module via the RAU control bus. Figure 5-1 gives a block diagram of the SMM.

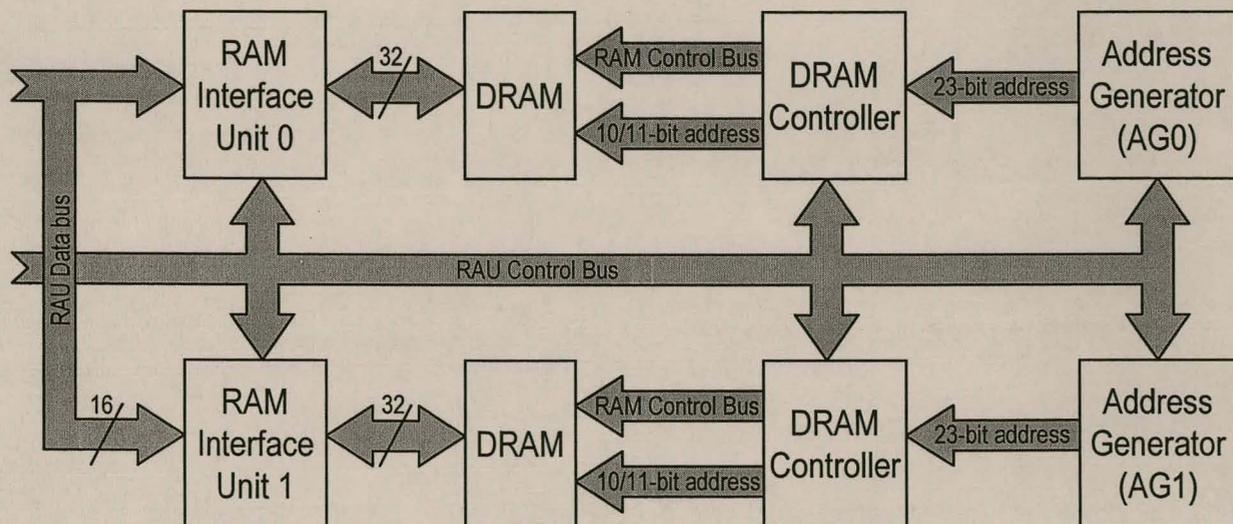


Figure 5-1 Block diagram of the system memory module (SMM)

Data is transferred to the RAU data bus from the DRM and saved in one of the two DRAM blocks shown in Figure 5-1. Each DRAM block can capture data at a maximum rate of approximately 30.8 Mbytes/s when writing 32 bits at a time. This rate is, however, not sustainable due to the fact that DRAM has to be refreshed periodically. For this reason the capture rate of the DRAM was established as 10 Mbytes/s. The capture rate of the SMM as a whole can be doubled to 20 Mbytes/s by implementing the two memory modules in parallel

and writing simultaneously to the two DRAM blocks. An isolated probe transmits data to the RAU at a rate of 10 Mbytes/s. The SMM can therefore capture the incoming data from the two isolated probes in real-time.

Two *RAM Interface Units* (RIUs) provide the interface between the 16-bit data bus and the 32-bit DRAM blocks, as shown in Figure 5-1. These devices are controlled from the RAU control module via the RAU control bus. The DRAM is implemented with standard Single In-line Memory Modules (SIMMs). Writing, reading and refreshing of the memory is managed by the *RAM controllers*, while the *address generators* supply the addresses that are to be read from or written to in the memory. The *RAM controllers* and *address generators* are controlled from the RAU control module via the RAU data bus.

## 5.2 Dynamic random access memory (DRAM)

There are a number of reasons why dynamic RAM (DRAM) is used as storage medium rather than static RAM (SRAM). SRAM stores data bits in an array of flip-flops, whereas DRAM stores data in charged capacitors that have to be refreshed periodically. SRAM is simpler to implement because no refresh clocks or timing complexities are involved. DRAM chips, however, occupy much less space than SRAM. DRAM is also substantially cheaper than SRAM. Consequently, it was decided to implement DRAM as the storage medium.

### 5.2.1 Read/write cycle

Figure 5-2 shows the read- and write-cycle timing for dynamic RAM.

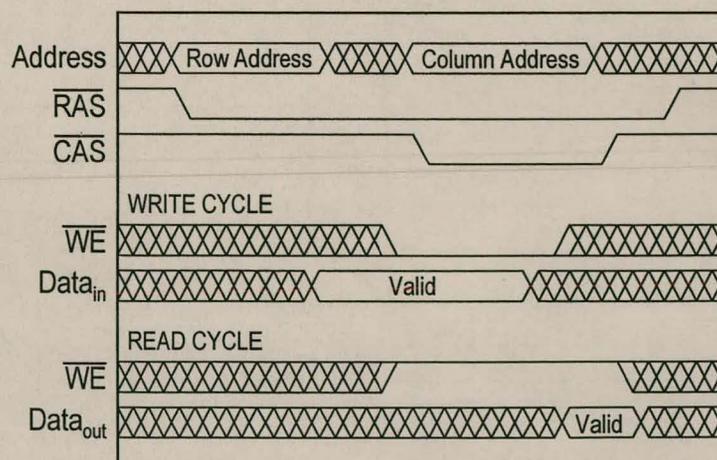


Figure 5-2 DRAM read and write cycles [27]

The address to be read from or written to is split into two halves that are multiplexed to the DRAM. With this method the number of address pins for addressing a memory device is halved. One half of the address, called the *row address*, is first latched into memory on the negative edge of *row address strobe* (RAS'). The other half, namely the *column address*, is then latched into memory on the negative edge of *column address strobe* (CAS').

Upon assertion of CAS' the data is written to or read from the DRAM, depending on the level of WE'. If WE' is high the data in the located memory address is transferred onto the data bus of the DRAM. If it is low, the data currently on the bus is written into the located memory address. Once the read/write operation is complete, CAS' is returned high where after RAS' is also returned high. RAS' must remain high for some minimum time (RAS-precharge time) before the next memory cycle can be started. The minimum time that a read/write cycle can be completed in is called the cycle-time.

### 5.2.2 Refresh cycle

A refresh cycle is similar to the read/write cycle as can be seen if Figure 5-3.

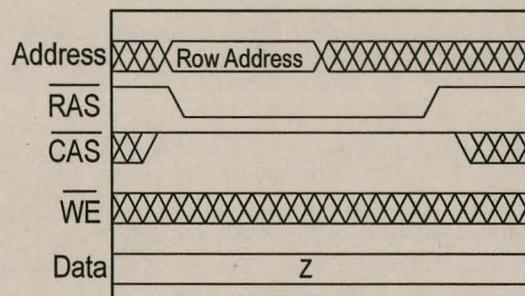


Figure 5-3 DRAM refresh cycle timing [27]

A complete row is refreshed at a time in DRAM. Only the *row address* therefore needs to be latched to the DRAM. The selected row is refreshed by asserting RAS' for the refresh period and keeping CAS' high during this time.

### 5.2.3 Implementation

Industry-standard single in-line memory modules (SIMMs) with 130 ns cycle time are used for implementing the DRAM blocks. This type of packaging has the following advantages:

- SIMMs are cheaper than individual RAM chips.

- The board space that a SIMM consumes is much less than that which individual RAM chips would consume.
- Because different sized SIMMs are pin-compatible, the size of the SMM can easily be adjusted by plugging in a larger or smaller SIMM.

It was determined in chapter 2 that the minimum sufficient storage capacity for the data acquisition system is 200 kbytes. At the time of design, the smallest commercially available DRAM SIMMs contained 4 Mbytes of storage space. Subsequently, the system has been designed to accommodate 4-Mbyte SIMMs, and also 8-Mbyte and 16-Mbyte SIMMs for cases where a larger storage capacity might be required. Each DRAM block accommodates up to two SIMMs per block. Appendix F gives detail block diagrams of the architecture of these SIMMs, including the timing parameters for the refreshing and read/write cycles.

### 5.3 RAM interface units (RIUs)

The RAM interface units (RIUs) provide the interface between the 16-bit RAU data bus and the 32-bit data lines of the DRAM blocks. When writing data to one of the DRAM blocks, the relevant RIU latches in two 16-bit bytes from the data bus and transfers the resulting 32-bit word to the DRAM module. The process is reversed when reading data from the DRAM. In this case 32-bit words are latched in from the DRAM data lines and transferred to the RAU data bus as two 16-bit bytes.

Figure 5-4 shows a block diagram of an RIU and the various control signals that control the operation of these devices. The *direction* (DIR) signal determines in which direction data flows. When writing data into the memory, DIR is set high. The 16-bit data on the RAU data bus is latched into *buffer A* of the RIU and transferred to *buffer B* and *buffer C* to be output to the DRAM. When reading data from the memory (DIR = 0), data is latched into *buffer B* and *buffer C* from the DRAM side. The data is then transferred to the RAU data bus via *buffer B*. The operation of the RIUs for reading/writing to the memory is discussed further in the following subsections.

#### 5.3.1 Writing to DRAM



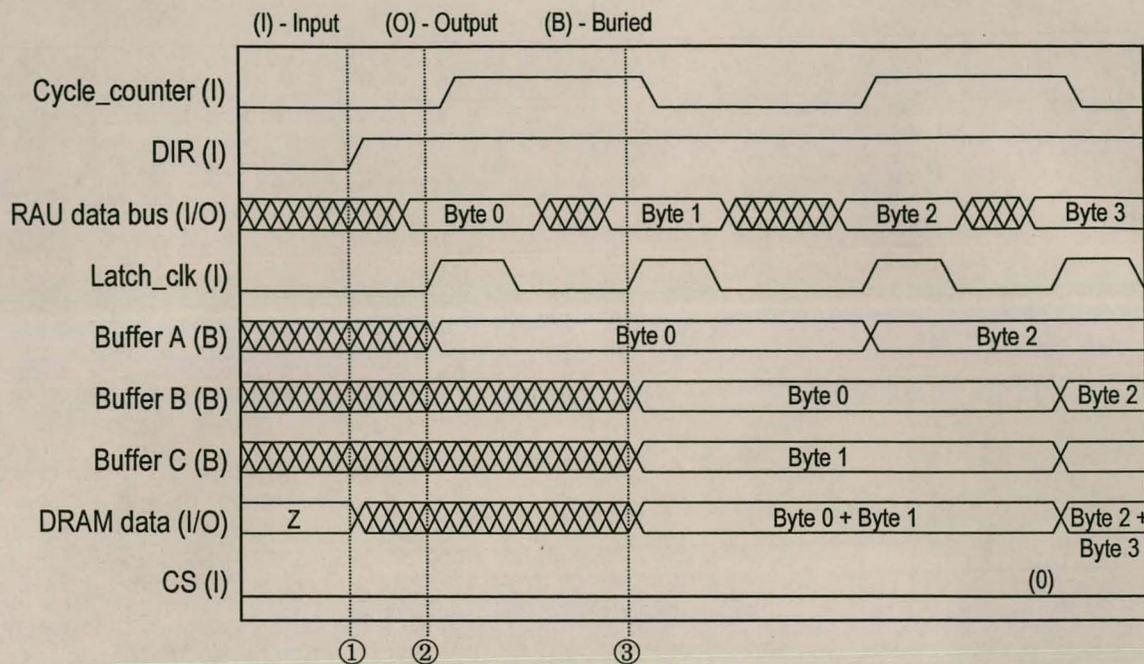


Figure 5-5 Timing waveforms for RIU0 when writing data into the DRAM

is driven high (①), enabling the tri-state buffers on the right in Figure 5-4. The 16-bit data on the RAU data bus is latched into either *buffer A* or *buffer C* of the RIU on the rising edges of *Latch\_clk*. If the *cycle\_counter* input is '0' on the positive edge of *Latch\_clk* (②), the data is latched into *buffer A*. With the next rising edge of *Latch\_clk* (③), *cycle\_counter* has been set high by the RAU control module (see chapter 6). The data on the RAU data bus is then latched into *buffer C*. At the same time the previously latched byte in *buffer A* is transferred to *buffer B*. At this stage the two 16-bit bytes (Byte 0 and Byte 1) appear at the 32-bit data inputs of the DRAM, ready to be written into memory.

The RAU control module uses the *chip select* (CS) signal to select between the two RIUs in the SMM. If CS is '0' RIU 0 can be accessed as shown in Figure 5-5, while RIU 1 can be accessed when CS is '1'.

### 5.3.2 Reading from DRAM

When reading data from the DRAM SIMMs, 32-bit words from the memory are latched into the active RIUs and transferred to the RAU data bus as two 16-bit bytes. Figure 5-6 shows the timing waveforms for RIU 0 when reading data from the DRAM.

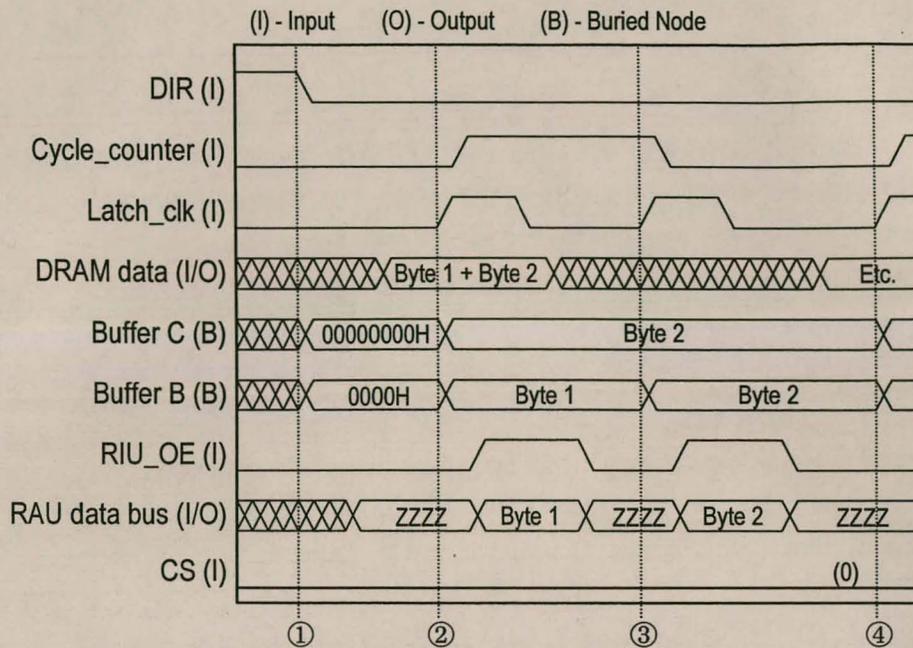


Figure 5-6 Timing waveforms for RIU0 when reading from the DRAM

The DIR input is pulsed low (①) by the RAU control module, indicating that data is to be transferred from the DRAM onto the RAU data bus. When a valid 32-bit word is present on the DRAM data bus, Latch\_clk is clocked twice from the RAU control module. On the first positive edge of Latch\_clk (②), the cycle\_counter input is low, causing the 32-bit word to be latched into the RIU. The most significant 16 bits are latched into *buffer B* and the least significant 16 into *buffer C*. The RAU control module then drives cycle\_counter high. On the second rising edge of Latch\_clk (③) the cycle\_counter is detected as high, causing the 16-bits currently in *buffer C*, to be latched into *buffer B*.

The 16-bit data in *buffer B* can be transferred to the RAU data bus by controlling the *chip select* (CS) and *output enable* (RIU\_OE) lines shown in Figure 5-4. CS determines whether RIU 0 or RIU 1 is accessed, as discussed in the previous section. When RIU\_OE is asserted, the data in *buffer B* of the selected RIU is transferred to the RAU data bus.

### 5.3.3 Implementation

The RIUs are implemented using two EPM7128SLC-7 erasable programmable logic devices (EPLDs) from *Altera*. The schematic diagram and VHDL-code of the RIUs are given in Appendix C and D respectively.

## 5.4 DRAM controllers

The DRAM controllers manage the read, write and refresh cycles of the respective DRAM modules via the RAM control bus shown in Figure 5-1. They are controlled from the RAU control module and receive the DRAM addresses from the address generators also shown in Figure 5-1. The DRAM controllers are implemented in EPM7128SLC-4 EPLDs from *Altera* [12]. The schematic diagram and VHDL-code are again given in Appendix C and D respectively.

Figure 5-7 shows a block diagram illustrating the architecture of a DRAM controller.

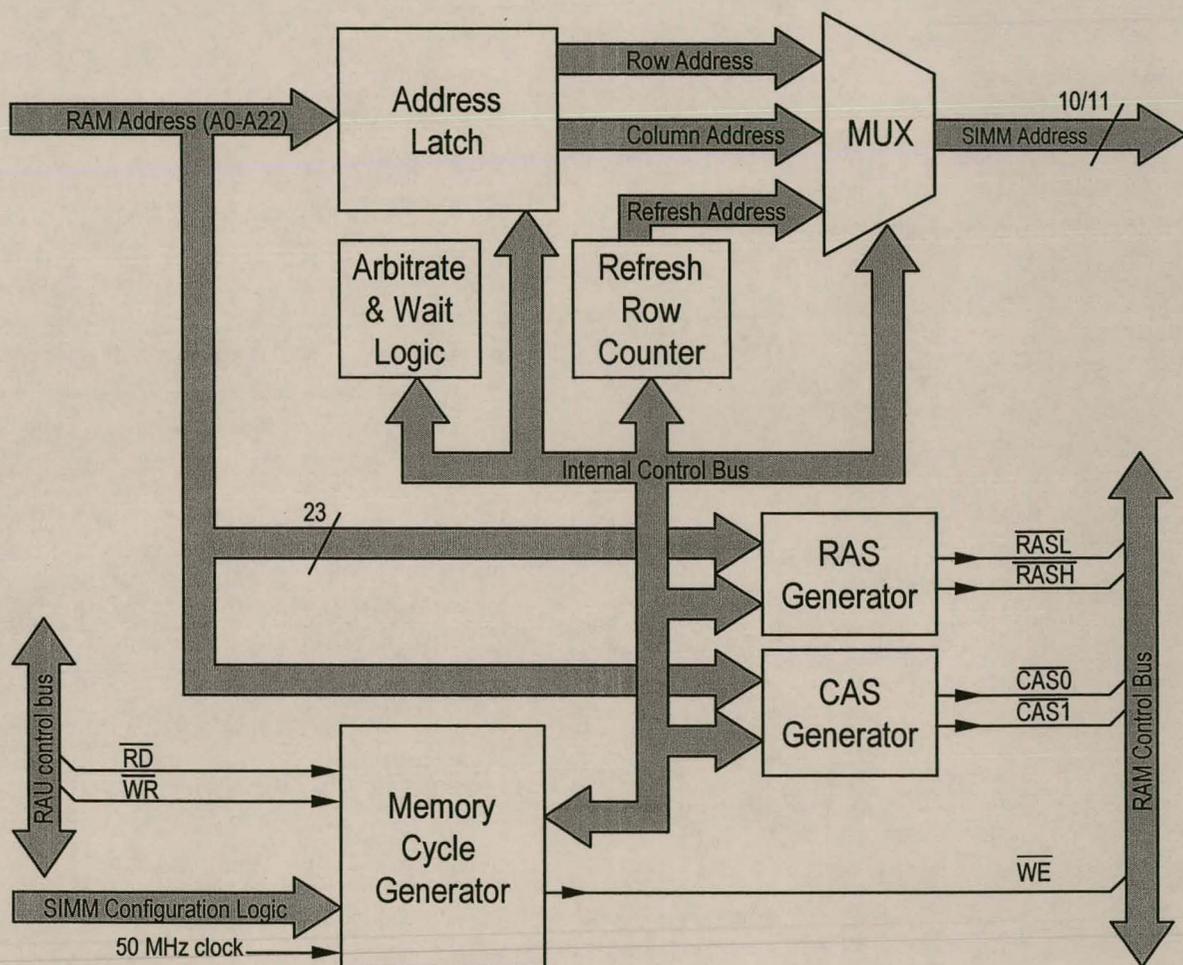


Figure 5-7 Block diagram of the DRAM controller

The *memory cycle generator* executes the read/write cycles and keeps track of when a refresh cycle needs to be executed. Read cycles are initiated if the *read* ( $RD'$ ) signal from the RAU control module is asserted. If the *write* ( $WR'$ ) signal is asserted a write cycle is initiated. The memory position in the DRAM must first be accessed before a read/write cycle can be

executed. The *memory cycle generator* latches the memory address from the *address latch* through to the DRAM as a row address and column address. The RAS/CAS generators then manipulate the RAM control signals shown in Figure 5-7 so that data is read from or written into the correct memory location. The *write-enable* (WE') output is used to distinguish between a read and a write command. If WE' is asserted (low), write commands can be executed, while reading from the memory is enabled when WE' is high.

#### 5.4.1 Refreshing

Each row in the DRAM SIMMs has to be refreshed periodically to retain the stored data. For the 4- and 8-Mbyte SIMMs each of the 1024 rows must be strobed with RAS' once every 16 ms [28], meaning that a row must be refreshed at least every 15.625  $\mu$ s (16 ms/1024). Each of the 2048 rows in a 16-Mbyte SIMM have to be refreshed once every 32 ms [28]. In this case a row must also be refreshed at least every 15.625  $\mu$ s (32 ms/2048). A row is refreshed every 1.420  $\mu$ s in the developed system. The *memory cycle generator* block contains an internal counter that determines when a refresh is required. The refresh row address is then multiplexed to the RAM and the relevant RAS-line is strobed. After completion of the refresh-cycle the *refresh row counter* is incremented, ready for the next cycle.

The *arbitrate & wait logic* block in Figure 5-7 ensures that read/write cycles and refresh cycles never overlap. When a negative edge is detected on either RD' or WR', the *arbitrate & wait logic* determines whether a refresh is currently in progress. If so, the refresh cycle is completed and the data cycle is attended to. If no refresh was in progress, the *arbitrate & wait logic* instructs the *memory cycle generator* to immediately execute the next refresh cycle. There after the requested data cycle is executed. This ensures that the read/write command is executed and that the refreshing continues within the required 15.625  $\mu$ s time slots.

#### 5.4.2 SIMM configuration logic

The *SIMM configuration logic* consists of a number of jumpers that have to be set according to the specifications of the DRAM SIMMs. In the schematic diagram of the RAU in Appendix C it can be seen where these configuration jumpers are to be set. Table 5-1 shows how the DRAM controllers interpret the various configuration inputs.

Table 5-1 Configuration logic for the DRAM controller

CONTROL SIGNAL NAME	FUNCTION	INTERPRETATION BY RAM CONTROLLERS
Present	Indicates number of SIMMs that is present.	Low – 1 SIMM High – 2 SIMMs
Auto	Enables the automatic detection of the SIMM sizes by using the SIMM outputs PD(0) and PD(1).	Low – Auto detection of SIMM sizes from PD(0) and PD(1). High – Manually setting of SIMM sizes with Size(0) and Size(1) jumpers.
Size (0-1)	Set SIMM sizes manually with these inputs.	00 – 4-Mbyte SIMM(s) 01 – 8-Mbyte SIMM(s) 10 – 16-Mbyte SIMM(s)
PD (0-1)	These are outputs from the SIMMs that indicate the sizes of the SIMMs.	11 – 4-Mbyte SIMM(s) 00 – 8-Mbyte SIMM(s) 10 – 16-Mbyte SIMM(s)

A DRAM controller can accommodate one or two DRAM SIMMs of sizes 4, 8 or 16 Mbytes. The number of SIMMs is set with a jumper that produces the Present input to the DRAM controllers. The size of the SIMM(s) can be set manually with the Size jumpers, or detected automatically on the *presence detect* (PD) outputs of the SIMM(s).

### 5.4.3 Address decoding

Four and eight megabyte SIMMs both have 10 address lines through which the row and column addresses are input. If SIMMs of these sizes are used in the SMM, the 10 least significant bits of the 23-bit DRAM address, i.e. A0-A9, are supplied as *row addresses* to the SIMMs, and A10-A19 as *column addresses*. A20 is used to determine whether the memory in SIMM0 or SIMM1 is to be accessed. The RAS' and CAS' inputs to the two SIMMs are controlled separately, and enable the DRAM controller to access each DRAM SIMM individually. Section 5.4.4 discusses the generation of the RAS' and CAS' signals in further detail.

With four megabyte SIMMs, DRAM chips are soldered only on the one side of each SIMM circuit board. Eight megabyte SIMMs, however, have DRAM chips soldered on both sides of the SIMM boards. If 8-Mbyte SIMMs are implemented, A21 of the address input is used to distinguish between the memory on the two sides of the SIMM circuit boards. The RAS' and CAS' signals are again used to distinguish between the two sides of these SIMMs

Sixteen megabyte SIMMs require 11 address inputs. With these SIMMs A0-10 is supplied as the *row addresses* and A11-A21 as the *column addresses*. The RAS/CAS generators then monitor A22 to distinguish between the memory in SIMM0 and SIMM1, and generate the RAS' and CAS' signals accordingly.

#### 5.4.4 RAS and CAS generators

The switching of the DRAM control signals for reading, writing and refreshing of the memory was discussed in section 5.2.1 and 5.2.2. The control signals used to perform these functions are the RAS' and CAS' signals, as shown in Figure 5-5 and Figure 5-6. Table 5-2 gives a list of the RAS' and CAS' outputs of the DRAM controller that drive the RAS/CAS inputs of the various SIMMs.

Table 5-2 Selection and access to the DRAM SIMMs with the RAS and CAS signals

SIMM Size	SIMM0/ SIMM1	Side0/ Side1	RAS Line & Address decoding		CAS Line & Address decoding	
4 Mbyte	SIMM0	-	RASL'	-	CAS0'	A20 = 0
4 Mbyte	SIMM1	-	RASL'	-	CAS1'	A20 = 1
8 Mbyte	SIMM0	0	RASL'	A20 = 0	CAS0'	A21 = 0
8 Mbyte	SIMM0	1	RASH'	A20 = 1	CAS0'	A21 = 0
8 Mbyte	SIMM1	0	RASL'	A20 = 0	CAS1'	A21 = 1
8 Mbyte	SIMM1	1	RASH'	A20 = 1	CAS1'	A21 = 1
16 Mbyte	SIMM0	-	RASL'	-	CAS0'	A22 = 0
16 Mbyte	SIMM1	-	RASL'	-	CAS1'	A22 = 1

Each DRAM controller can accommodate up to two SIMMs, as was mentioned in the previous section. The CAS-outputs of the controller are used to distinguish between the two SIMMs (SIMM0 and SIMM1) when accessing the memory. CAS0' and CAS1' are generated by the *CAS generator* in Figure 5-7, and respectively drive the CAS-inputs of SIMM0 and SIMM1.

With 4 and 16-Mbyte SIMMs, DRAM chips are soldered only on the one side of each SIMM circuit board. The RASL' output of the DRAM controller then drives the RAS-inputs of the SIMMs. 8-Mbyte SIMMs, however, have DRAM chips soldered on both sides of the SIMM circuit boards. In such a case RASL' and RASH' distinguish between the DRAM chips on the two sides of the SIMM(s).

The *memory cycle generator* in Figure 5-7 informs the *RAS* and *CAS* generators when a read/write or refresh cycle has to be performed. The *RAS/CAS* generators then check the RAM address inputs A0-A22 to determine which of RASL', RASH', CAS0' and CAS1' is to be asserted.

## 5.5 DRAM address generator

Two DRAM address generators are implemented in order to supply the 23-bit addresses to the DRAM controllers. *Altera* EPLDs are used to implement the devices (VHDL-code [12] given in Appendix D). The address generators are controlled from the RAU control module via the RAU control bus. Figure 5-8 gives a block diagram illustrating the operation of the DRAM address generators.

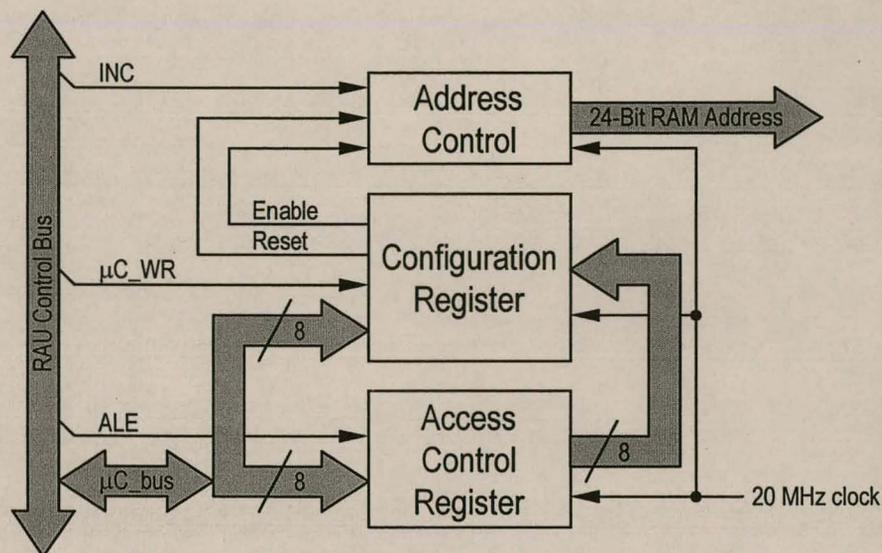


Figure 5-8 Block diagram of the RAM address generator

The *address control* block in Figure 5-8 controls the 23-bit address that is output by the address generator. The *address control* utilises the *increment* (INC) signal from the RAU control bus and the internal Reset and Enable signals from the *configuration register*. The *configuration register* is an 8-bit register of which Enable and Reset are the two least significant bits. If the Enable bit is high, the 23-bit address is incremented each time a positive edge on INC is detected. The address, however, remains constant on these edges if Enable is low. When Reset is high the output address is reset to zero.

The *configuration registers* in the two DRAM address generators are configured with the aid

of the *access control registers* in the devices, and the  $\mu\text{C\_bus}$  (8 bits), ALE and  $\mu\text{C\_WR}$  signals shown in Figure 5-8. The  $\mu\text{C\_bus}$ , ALE and  $\mu\text{C\_WR}$  signals are sent from the host interface module in the RAU via the RAU control bus. In chapter 7 it is discussed how these signals are generated. Note, however, that these signals are inputs to both the DRAM address generators.

The data on  $\mu\text{C\_bus}$  is latched into the *access control registers* of both address generators (AG0 and AG1) on every positive edge of ALE. The data in the *access control registers* then determine whether the *configuration registers* in the two address generators are to be configured or not. If the data in the *access control register* of AG0 is equal to *10100000*, the 8-bit data on  $\mu\text{C\_bus}$  is latched into the *configuration register* of AG0 on the rising edges of  $\mu\text{C\_WR}$ . The data on  $\mu\text{C\_bus}$  is in turn latched into the *configuration register* of AG1 on the rising edges of  $\mu\text{C\_WR}$ , if the *access control register* in AG1 contains a byte equal to *10101111*. The two configuration registers can therefore be configured with the same control signal lines i.e.  $\mu\text{C\_bus}$ ,  $\mu\text{C\_WR}$  and ALE via the RAU control bus.

## Chapter 6

### RAU control module

#### 6.1 Introduction

The remote acquisition unit (RAU) consists of four modules, namely the data receiving module (DRM), the system memory module (SMM), the host interface module (HIM) and the RAU control module. The DRM and SMM have been discussed in the previous two chapters. In this chapter the implementation and operation of the RAU control module is discussed. Chapter 7 then discusses the host interface module together with the software environment from where all data acquisition hardware is controlled.

The RAU control module provides the required control signals to the DRM and the SMM and then monitors the operation of these modules. Furthermore, it sends the required conversion clock signals that were discussed in section 3.5 to the two isolated probes. Figure 6-1 recaptures the RAU in block diagram format to illustrate the positioning of the control module relative to the other system components.

Firstly, the control module provides the sampling clocks for the ADCs on the isolated probes during an acquisition cycle. The sampling clocks are generated in the control module and transferred to the fibre optic transmit logic on the RAU. The clock signals are then transmitted across the 155-MBd fibre optic control links to the isolated probes. Once the acquisition cycle is complete a logical low is sent across the control links. This effectively disables further transmission of data from the isolated probes.

The control module also provides control signals to the data receiving module (DRM) and the system memory module (SMM) on the RAU. Regarding the DRM, the control module is responsible for monitoring when this module has received valid data. In such a case the data must be transferred to the RAU data bus in the correct format and with the correct timing

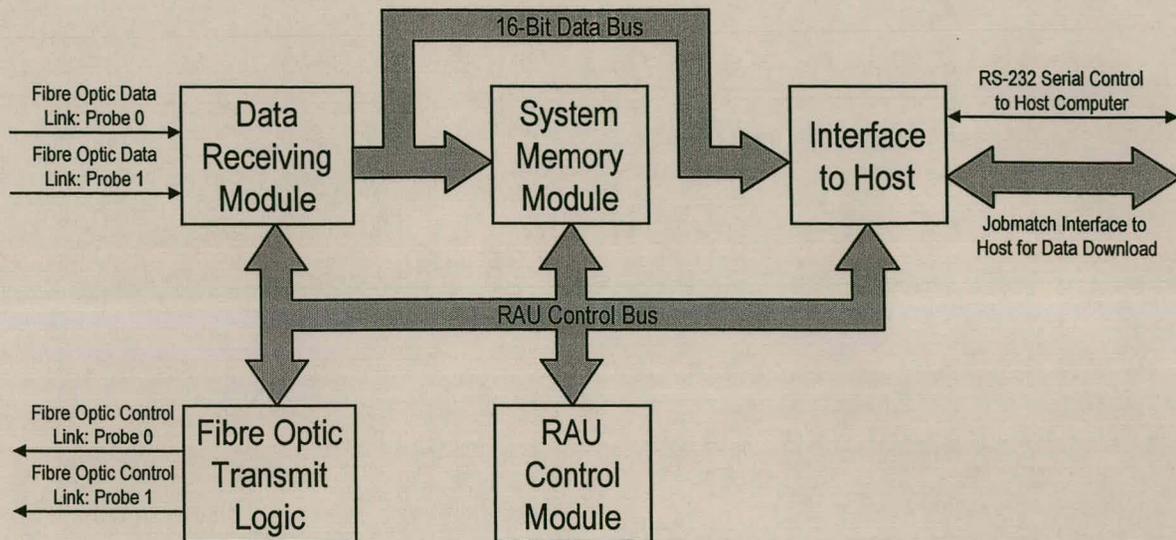


Figure 6-1 Remote acquisition unit block diagram

parameters relative to the SMM. Active control of the DRM is required only when data is being transmitted from the isolated probes to the RAU.

If the complete acquisition system is implemented, the SMM contains six devices that are controlled from the RAU control module. These are: two RAM interface units (RIUs), two RAM controllers and two RAM address generators. During data acquisition, the RIUs are controlled to latch in data from the RAU data bus. The address generators and RAM controllers are then controlled from the control module so that the data is written into the SMM memory at the correct address locations. The control module must ensure that proper synchronisation between these components is maintained to ensure that no data is corrupted or lost. When downloading the SMM data to the host computer, the RAM address generators and RAM controllers sequentially read the data in each memory position in the SMM. The RIUs latch in the 32-bit words read from the memory and transfer the data to the 16-bit RAU data bus.

Finally, the RAU control module interfaces to the host interface module. This module is used when configuring the RAU from the host PC and when downloading data to the host. Configuration of the RAU control module involves the setting of the operating mode and operating specifications for the system in this module. The user can therefore control the acquisition system from a remote location via the host interface module.

This chapter discusses the implementation of the RAU control module. In section 6.2 the RAU control bus is analysed with reference to the various system components and their control signals. The hardware implementation is discussed in section 6.3.

## 6.2 RAU control bus

In the previous three chapters the implementation of the isolated probes, the data receiving module and the system memory module were discussed. The host interface module is reviewed in the first part of chapter 7. The implementation of these four modules is discussed with reference to the control signals that manage their operation. These control signals are either supplied from one of the other above-mentioned modules, or in most cases by the RAU control module. The interconnecting control signals between the various system components on the RAU constitute the RAU control bus, as shown in Figure 6-2.

The serial-to-parallel decoders supply control signals to the FIFOs/demuxes, as discussed in chapter 4. The P0-CKR and P1-CKR clock signals indicate the rate at which bytes are decoded and output by the respective decoders. P0-SC/D' and P1-SC/D' give an indication on whether the bytes are data or control bytes. If P0-RVS or P1-RVS is asserted, a corrupted byte has been received from probe 0 or probe 1 respectively.

The RAU control module functions with a 50 MHz clock signal (*mclk*) that is supplied from a crystal oscillator on-board the RAU. The control module supplies control signals to the *FIFOs/demuxes*, the *RIUs*, the *RAM controllers*, the *RAM address generators* and the *fibre optic transmit logic*. It is in turn configured from the *host interface module*.

The FIFOs/demuxes for probe 0 and probe 1 are controlled individually with a set of signals, i.e. Reset, P<sub>x</sub>-Error, *output enable* (P<sub>x</sub>-OE') and *data ready* (P<sub>x</sub>-DRDY), where x is 0 or 1. Reset is used to initialise these devices, whereas P<sub>x</sub>-Error informs the control module when data from the respective isolated probes has been lost or corrupted before being saved. The P<sub>x</sub>-DRDY signals indicate when the FIFOs/demuxes contain data that is ready to be transferred to the RAU data bus. By asserting the P<sub>x</sub>-OE' lines, the data from the devices is transferred to the RAU data bus. It is essential that only one P<sub>x</sub>-OE' line is enabled at a time to avoid bus contention.

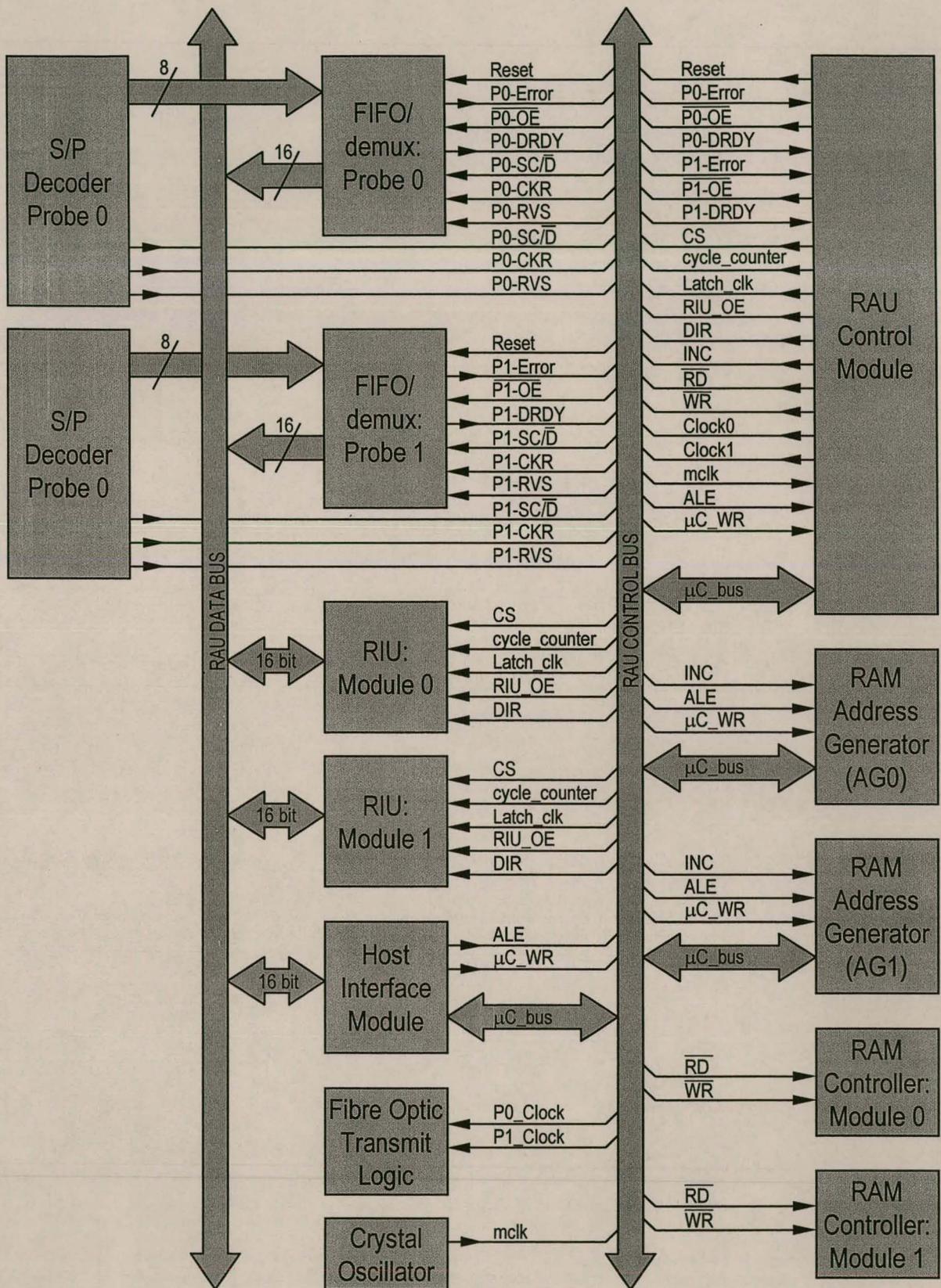


Figure 6-2 Remote acquisition unit control and data bus

The implementation of the RIUs is discussed in chapter 5. These devices are controlled from the RAU control module via the *chip select* (CS), *cycle\_counter*, *Latch\_clk*, *output enable* (RIU\_OE) and *direction* (DIR) signals. During data acquisition, data is latched from the RAU data bus into the RIUs and transferred to the DRAM modules (see chapter 5). This data is then read back onto the RAU data bus, through the RIUs, during *download mode*. The DIR signal determines in which direction the data flows (DIR = 1: from data bus to DRAM, DIR = 0: from DRAM to data bus). Depending on the value of CS, either RIU 0 or RIU1 is activated. The positive edges of *Latch\_clk* are used to latch data into the RIUs, whether it be from the RAU data bus or from the DRAM. RIU\_OE is asserted when transferring data from the selected RIU (as determined by CS) to the RAU data bus.

When writing into memory, pairs of 16-bit bytes are latched into the RIUs and output as 32-bit words to the DRAM. The *cycle\_counter* signal in Figure 6-2 indicates whether the first or second byte of a pair has been latched into an RIU. When read from memory, 32-bit words are latched into the RIUs and transferred to the RAU data bus as two 16-bit bytes. Again *cycle\_counter* indicates to the RIUs if one or two of the 16-bit bytes have been transferred to the data bus.

Regarding the RAM address generators, the *increment* (INC) signal is provided from the RAU control module. A positive edge on INC will increment the address that is currently at the output of the address generators. It was mentioned in chapter 5 that data is written simultaneously to DRAM module 0 and DRAM module 1. For this reason the two address generators are controlled in parallel, and the same INC signal is sent to both address generators.

The RAM controllers are also controlled in parallel as explained above. Positive edges on *read* (RD') and *write* (WR') respectively cause data to be read from or written to the DRAM SIMMs. The control module must ensure that the correct timing specifications, i.e. set-up and hold times are maintained between the various DRAM components.

The RAU control module and the RAM address generators are configured from the host PC via the host interface module. The implementation and operation of this module is discussed in detail in chapter 7. The signals supplied from the module are an 8-bit data bus ( $\mu\text{C\_bus}$ ) and two control signals namely *address latch enable* (ALE) and *write* ( $\mu\text{C\_WR}$ ). The operating

mode and operating specifications for the system are programmed into the RAU control module and address generators with these signals.

Finally, during data acquisition, the sampling clocks for the isolated probe ADCs are sent from the RAU control module to the *fibre optic transmit logic*. This logic then transfers the clocks across the 155-MBd fibre optic control links to the two isolated probes.

### 6.3 Implementation and operation of the RAU control module

A single field programmable gate array (FPGA) manufactured by *Altera* is implemented for realisation of the RAU control module. The FLEX10K10LC84-4 FPGA is used, as it contains sufficient I/O pins, logic cells and speed requirements for performing the required functions of the control module. The schematic diagram representation is contained in Appendix C.

The FPGA is implemented with *Verilog* hardware description language (VHDL), of which the code is given in Appendix D. Figure 6-3 gives a high-level block diagram of the device. The control module provides the sample clocks to the *fibre optic transmit logic* and control signals to the *FIFOs/demuxes* (in DRM), *RIUs* (in SMM), *RAM controllers* (in SMM) and *RAM address generators* (in SMM). Each of the components in Figure 6-3 are connected together with the *control module bus*, and are controlled with the aid of a 50 MHz clock signal, *mclk*. The operation of the various components depends upon the *configuration logic*, which is programmed from the host PC via the *access control register*.

#### 6.3.1 Configuration of the RAU control module

The configuration logic in Figure 6-3 consists of two 8-bit *configuration registers* that respectively determine the operating mode and operating specifications of the data acquisition system. The operating mode information is contained in the *op\_mode* register while the operating specifications are stored in the *op\_specs* register. The *op\_mode* register is defined as shown in Table 6-1.

The two least significant bits (LSBs) determine the operating mode of the system. If *op\_mode.0* is high, the isolated probes are activated to sample the analog input signals. The

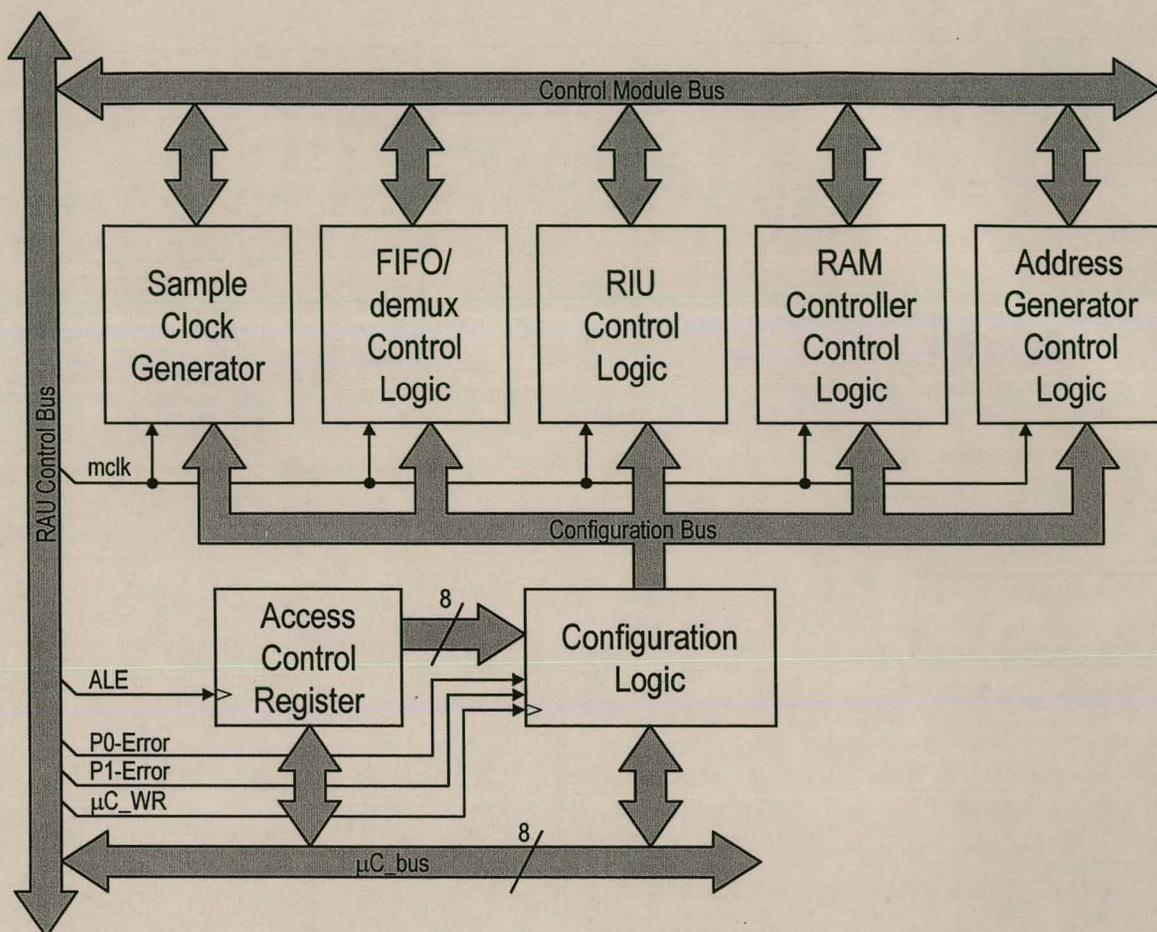


Figure 6-3 Block diagram of the RAU control module

Table 6-1 *Op\_mode* register in the RAU control module

Bit number	Interpretation
<i>op_mode.7</i> (MSB)	Reserved for future use.
<i>op_mode.6</i>	Reserved for future use.
<i>op_mode.5</i>	Reserved for future use.
<i>op_mode.4</i>	Reserved for future use.
<i>op_mode.3</i>	Reserved for future use.
<i>op_mode.2</i>	Reserved for future use.
<i>op_mode.1</i>	If this bit is high, and <i>op_mode.0</i> is low, the data in the SMM is downloaded to the host ( <i>download mode</i> ).
<i>op_mode.0</i> (LSB)	If this bit is high, the isolated probes are activated and the sampled data is stored in the SMM ( <i>acquisition mode</i> ).

data samples are transmitted to the RAU where they are saved in memory. This mode of operation is defined as *acquisition mode*. If *op\_mode.0* is low but *op\_mode.1* is high, the mode is set to *download mode*. In this case the isolated probes are deactivated and the data in

the SMM is transferred to the RAU data bus. If neither *op\_mode.0* nor *op\_mode.1* is set, the system switches to a default mode named *idle mode*. In the next sections of this chapter more detail is given on the various operating modes of the system.

The *op\_specs* register is shown in Table 6-2.

Table 6-2 *Op\_specs* register in the RAU control module

Bit number	Interpretation
<i>op_specs.7</i> (MSB)	'1' – Isolated probe 0 active. '0' – Isolated probe 0 inactive.
<i>op_specs.6</i>	'1' – Isolated probe 1 active. '0' – Isolated probe 1 inactive.
<i>op_specs.5</i>	'1' – DRAM module 0 active. '0' – DRAM module 0 inactive
<i>op_specs.4</i>	'1' – DRAM module 1 active. '0' – DRAM module 1 inactive
<i>op_specs.3</i>	Reserved for future use.
<i>op_specs.2</i>	Sample frequency bit 2.
<i>op_specs.1</i>	Sample frequency bit 1.
<i>op_specs.0</i> (LSB)	Sample frequency bit 0.

The data acquisition system topology contains two isolated probes and two parallel memory modules in the SMM (see Figure 5-1). It is, however, possible to enable only one of the isolated probes and the memory modules at a time. The two most significant bits (MSBs) in the *op\_specs* register determine which of the isolated probes are active. *Op\_specs.5* and *op\_specs.4* in turn indicate which of the memory modules are active. The control software discussed in chapter 7 checks that at least one isolated probe and one RAM module is activated at a time. The three LSBs in Table 6-2 contain information on the sampling frequency of the ADCs on the isolated probes. Table 6-3 illustrates how these three bits are interpreted.

Table 6-3 Specification of the sample frequency in the RAU control module

<i>Op_specs.2 – op_specs.0</i>	Sample frequency
000	10 kHz
001	50 kHz
010	100 kHz
011	500 kHz
100	1 MHz
101	5 MHz
other	5 MHz

The sample frequencies range from 10 kHz to 5 MHz, as shown above. The default sample frequency is 5 MHz.

The two configuration registers are configured from the host PC via the host interface module. The *access control register* in Figure 6-3 is an 8-bit register that enables the user to distinguish between the two configuration registers. Each configuration register has an 8-bit register number that must be loaded into the *access control register* before any data can be written to that register. The register numbers of the *op\_mode* and *op\_specs* registers are *00001010* and *00001111* respectively.

On positive edges of *address latch enable* (ALE), the data present on  $\mu\text{C}_{\text{bus}}$  is latched into the *access control register*. If this data corresponds to the register number of any of the configuration registers, the data on  $\mu\text{C}_{\text{bus}}$  is latched into that configuration register on the positive edges of  $\mu\text{C}_{\text{WR}}$ .

During *acquisition mode*, the *configuration logic* monitors the levels of P1-Error and P0-Error. If either of these signals switch high, data from the isolated probes has been corrupted or lost before being saved. In such a case the *op\_mode* register is set to *00000000*, causing the RAU control module to switch to *idle mode*.

### 6.3.2 Sample clock generator

The sample clocks for the isolated probes are generated in the RAU control module. The sampling frequency can be selected from one of six discrete frequencies between 5 MHz and 10 kHz in the prototype system. It would be possible to reprogram the RAU control module so that any frequency can be selected in this range. An 8254 could also be implemented to manage the clock generation functionality of the system. These approaches are, however, left to be explored as future options.

The RAU control module operates from a 50 MHz master clock *mclk* (see Appendix C). The sample clock generator in Figure 6-3 divides this clock down to obtain the different sample frequencies shown in Table 6-3. Figure 6-4 shows the architecture of the clock generator in the RAU control module.

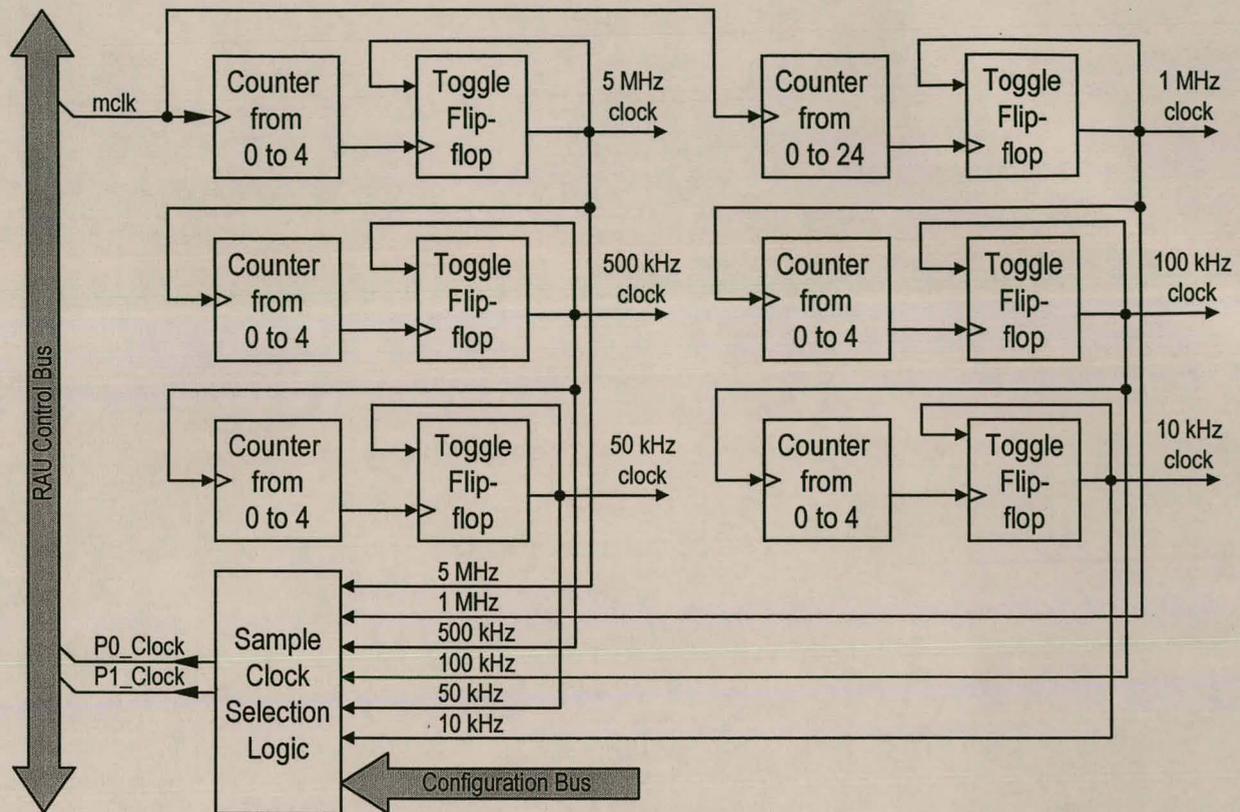


Figure 6-4 Sample clock generator block diagram

The various sample clocks are divided down from *mclk*, which is a 50 MHz crystal oscillator clock signal. The sample clocks are generated with the aid of counters and toggle flip-flops (T flip-flops). To generate a 5 MHz clock signal, a counter is incremented on each positive edge of *mclk*. After 5 clock cycles the counter reaches its maximum value of 4. This causes the counter to reset to zero and the T flip-flop to toggle. The resulting output of the T flip-flop is a 5 MHz square wave with 50 % duty cycle. The 5 MHz signal is used similarly to generate a 500 kHz signal, which in turn generates a 50 kHz signal as shown above.

Another counter is incremented on the positive edges of *mclk* for generating a 1 MHz clock signal. After 25 clock cycles the counter resets to zero and the connected flip-flop toggles. This produces a signal that toggles from '0' to '1' or '1' to '0' every 1  $\mu$ s, which is a 1 MHz square wave or clock signal. 100 KHz and 10 kHz sample clocks are derived from the 1 MHz signal as shown in Figure 6-4.

The *P0\_Clock* and *P1\_Clock* signals are transmitted to the fibre optic transmit logic. These signals depend on the operating mode and operating specifications of the system. The *sample*

*clock selection logic* monitors the *configuration logic* to obtain this information. If the system is in *acquisition mode* the *sample clock selection logic* reads the desired sampling frequency in the *op\_specs* register and outputs the correct sample clock signal at P0\_Clock and P1\_Clock. For operating modes other than *acquisition mode*, a logical zero is output at both P0\_Clock and P1\_Clock.

### 6.3.3 FIFO/demultiplexer control logic

The FIFO/demux control logic in the RAU control module monitors when valid data is present in the FIFOs/demuxes. In such a case the FIFOs/demuxes are controlled so that the data is transferred to the RAU data bus. The control of the FIFOs/demuxes depends on the operating mode and operating specifications of the data acquisition system.

The FIFOs/demuxes are controlled individually from the RAU control module with the control signals shown in Figure 6-2. A block diagram of the FIFO/demux control logic is given in Figure 6-5.

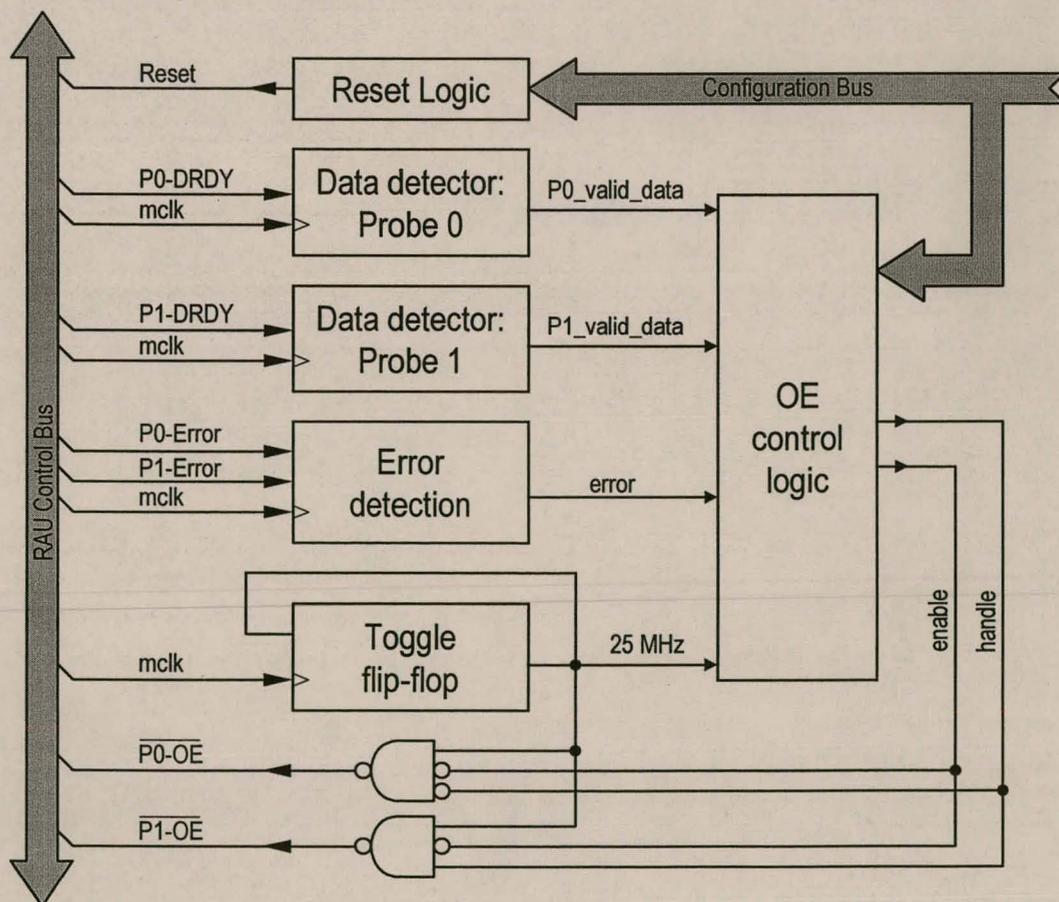


Figure 6-5 Control logic for FIFOs/demultiplexers in the RAU control module

The *data detectors* in Figure 6-5 sample the DRDY signals from the FIFOs/demuxes on every positive edge of *mclk*. If P0-DRDY or P1-DRDY switches high, the data in the relevant FIFO/demux has to be transferred to the RAU data bus. The *data detectors* then request the *OE control logic* to assert the relevant OE' line in order to accomplish this.

If only one of the isolated probes is active, the data from that probe's FIFO/demux is transferred to the RAU data bus as it is received by the FIFO/demux. If both probes are active, data is alternatively read from the two FIFOs/demuxes and transferred to the RAU data bus.

The Px-OE' signals are controlled with the aid of the two internal signals, *enable* and *handle*, and the *toggle flip-flop* in Figure 6-5. The *enable* signal must be asserted for either of the Px-OE' lines to be asserted. If *handle* is '0' then P0-OE' can be asserted, whereas *handle* must be '1' for P1-OE' to be asserted.

The output of the *toggle flip-flop* is a 25 MHz square wave signal. With the AND-gates implemented as shown in Figure 6-5, P0-OE' and P1-OE' can be asserted only in the 20 ns that the *toggle flip-flop* output is high. The *OE control logic* sets the levels of *enable* and *handle* in the 20 ns that the *toggle flip-flop* output is low. When the flip-flop output then switches high, one of P0-OE' or P1-OE' can be asserted, depending on the logic levels of *enable* and *handle*. With this configuration it is ensured that only one of the Px-OE' lines is asserted at any given time and therefore bus contention is avoided.

The *OE control logic* has to determine which of the isolated probes are active during data acquisition in order to control the *handle* signal. This information is obtained from the *configuration bus*. If only probe 0 is active then *handle* is switched to zero for the whole acquisition cycle. Likewise, it is set high if only probe 1 is active. If both probes are active, data is alternatively written from probe 0 and probe 1 onto the data bus. The *OE control logic* keeps the value of *handle* low until P0-OE' has been asserted for 20 ns. It then switches high until P1-OE' is asserted for 20 ns, etc.

The Error signals from the FIFOs/demuxes are sampled on every positive edge of *mclk*. If either P0-Error or P1-Error is asserted, it means that data from the relevant probe has been lost or corrupted. The *OE control logic* then pulses *enable* low, disabling further transfer of data

to the RAU data bus. As discussed in section 6.3.1, the operating mode of the system is set to *idle mode* when such an error occurs. As soon as *idle mode* is entered the *reset logic* asserts the Reset output to the FIFOs/demuxes, effectively disabling them.

### 6.3.4 RAM interface unit control logic

The RAM interface units provide the interface between the RAU data bus and the DRAM modules in the SMM. During *acquisition mode* data is transferred from the RAU data bus to the DRAM modules, while with *download mode* the data is transferred back to the RAU data bus. Figure 6-6 gives a block diagram of the RIU control logic in the RAU control module.

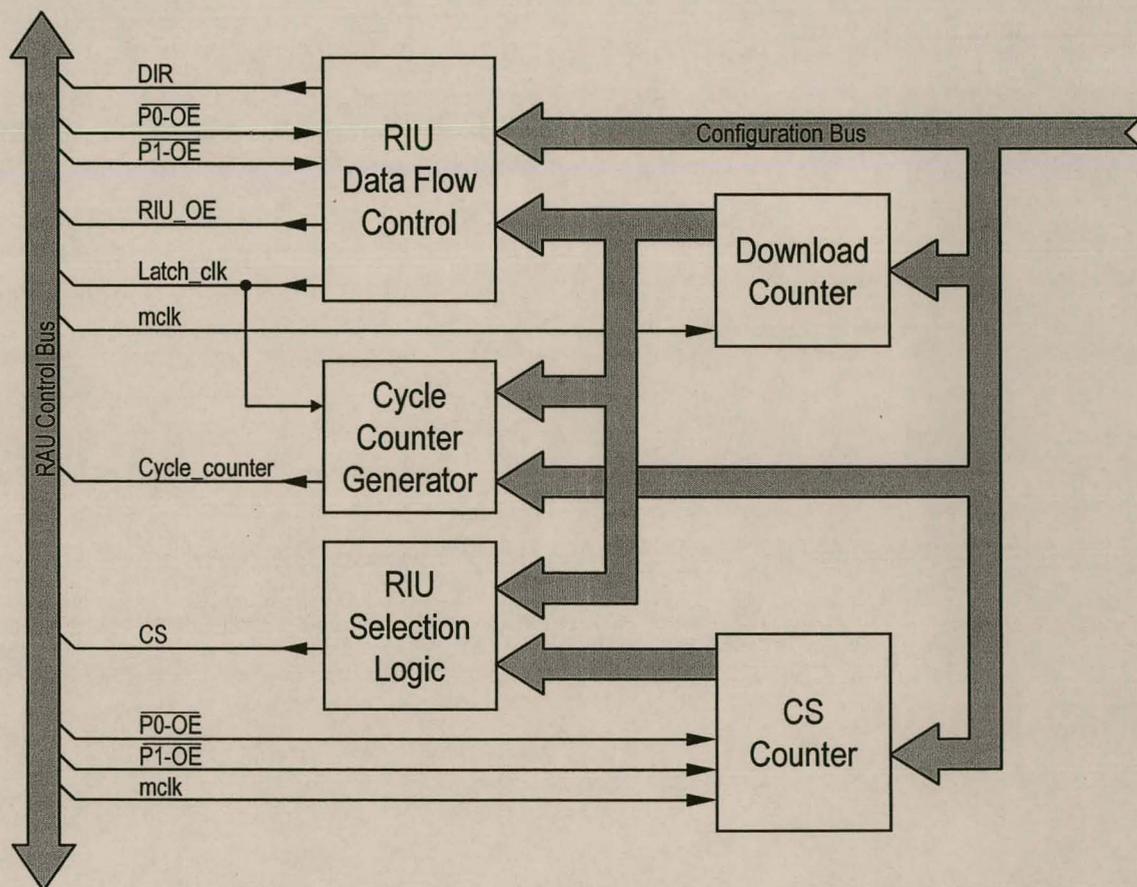


Figure 6-6 RAM interface unit control in the RAU control module

The *data flow control* block monitors the operating mode of the system, via the configuration bus, to determine the level of DIR. Together with the *cycle counter generator*, it controls the flow of data through the RIUs during *acquisition* and *download mode*. The *download counter* in Figure 6-6 controls the operation of the RIU control logic during *download mode*, and is discussed further in section 6.3.4.2. The *RIU selection logic* utilises the *download counter*

and the *CS counter* in order to control the CS inputs of the RIUs. The following subsections discuss the operation of the RIU control logic during acquisition mode and download mode.

### 6.3.4.1 Acquisition mode

Figure 6-7 shows a timing diagram that illustrates the operation of the RIU control logic during *acquisition mode*.

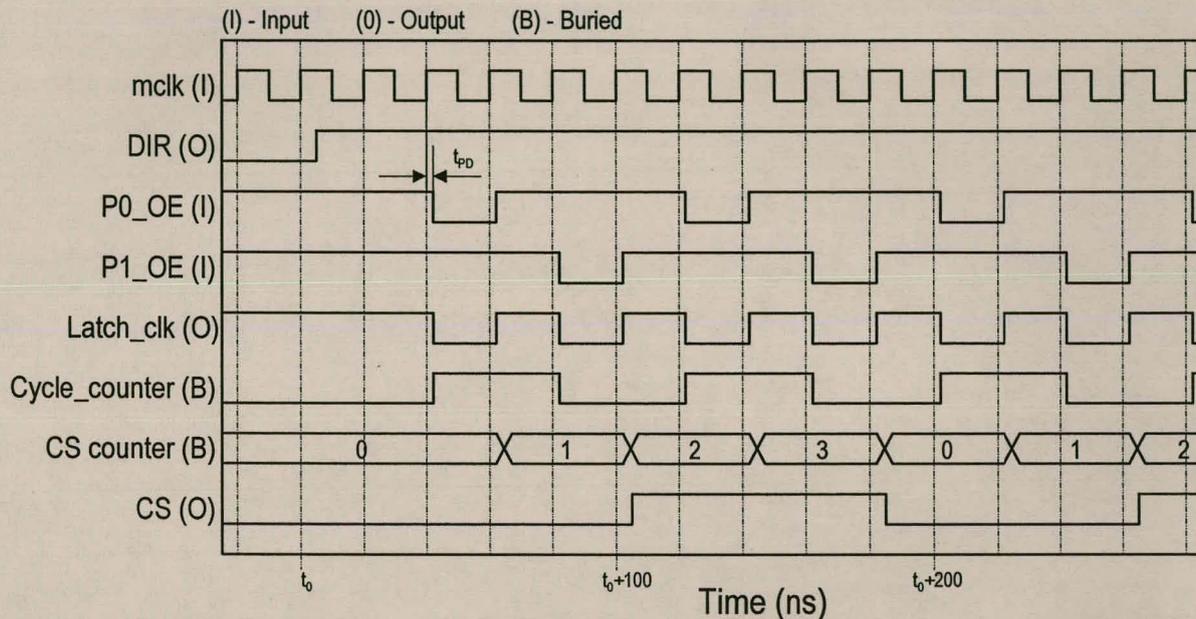


Figure 6-7 RAM interface unit timing waveforms during *acquisition mode* (both RAM modules active)

The *RIU data flow control* in Figure 6-6 sets the DIR input of the RIUs to a logical high. Each time a 16-bit data byte is transferred to the data bus from a FIFO/demux, the byte must be latched into an RIU. Such a data byte is present on the data bus when either P0-OE' or P1-OE' is asserted. When the asserted Px-OE' line switches high again, the data on the RAU data bus is still valid for a few nanoseconds. This is because of the propagation delay of the Px-OE' signal from the RAU control module to the relevant FIFO/demux and the settle and hold times of the FIFO/demux. Latch\_clk is pulsed high at the same time that the asserted Px-OE' line rises, and the data present on the RAU data bus is latched into an RIU. Latch\_clk is therefore programmed to pulse low when either P0-OE' or P1-OE' pulses low, and to remain high otherwise.

When two 16-bit bytes have been latched into an RIU they are output as a 32-bit word.

Cycle\_counter indicates whether one or two of the bytes have already been latched into the RIU. If cycle\_counter is '0' only one byte has been latched into the device, whereas two bytes have been latched in if cycle\_counter is '1'. Cycle\_counter is toggled on the negative edges of Latch\_clk to ensure that it is valid on the rising edge of Latch\_clk.

The CS signal is generated by the *RIU selection logic* in Figure 6-6. This signal determines into which RIU data is to be latched. A low or high on CS respectively accesses RIU 0 and RIU 1. If only one DRAM module is active, CS is permanently low, and RIU 0 is accessed. If both DRAM modules are active as in Figure 6-7, the *CS counter* in Figure 6-6 keeps track of the RIU selection. The counter is incremented each time a byte is transferred to the RAU data bus, in other words with every negative edge of either P0-OE' or P1-OE'. When the counter reaches a value of 3 it wraps around to zero again. The *chip selection logic* monitors the *CS counter*. If the counter value is 0 or 1, CS is driven low, while it is pulsed high for values of 2 and 3. The result is that two 16-bit bytes are latched into RIU 0 where after two bytes are latched into RIU 1. The process is repeated during the rest of the acquisition cycle.

#### 6.3.4.2 Download mode

DIR is pulsed low during *download mode* by the *RIU data flow control*. Two 16-bit bytes are output for every 32-bit word that is latched into the RIUs from the active DRAM modules. On the rising edges of Latch\_clk, the active RIUs check the level of cycle\_counter to determine what action must be taken. If cycle\_counter is low a new 32-bit word is latched into the active RIUs from the DRAM modules. The most significant 16 bits of the word can then be transferred to the RAU data bus by asserting RIU\_OE. If cycle\_counter is low on a rising edge of Latch\_clk, the least significant 16 bits of the 32-bit word are made available to the RAU data bus. By asserting RIU\_OE the 16-bit byte can be transferred to the data bus.

The *download counter* in Figure 6-6 controls the timing of the RIU control logic during this mode. It is incremented on each positive edge of mclk, from 0 to 99. When it reaches the value of 99, it returns to zero again. With mclk a 50 MHz clock signal, the *download counter* counts from 0 to 99 every 2  $\mu$ s. These 2  $\mu$ s-cycles are used for transferring data from the DRAM modules through the RIUs to the RAU data bus. Figure 6-8 shows the timing waveforms.

At  $t = t_0$  the value of the *download counter* is 0. At  $t = t_0+0.5$  microseconds the counter value is

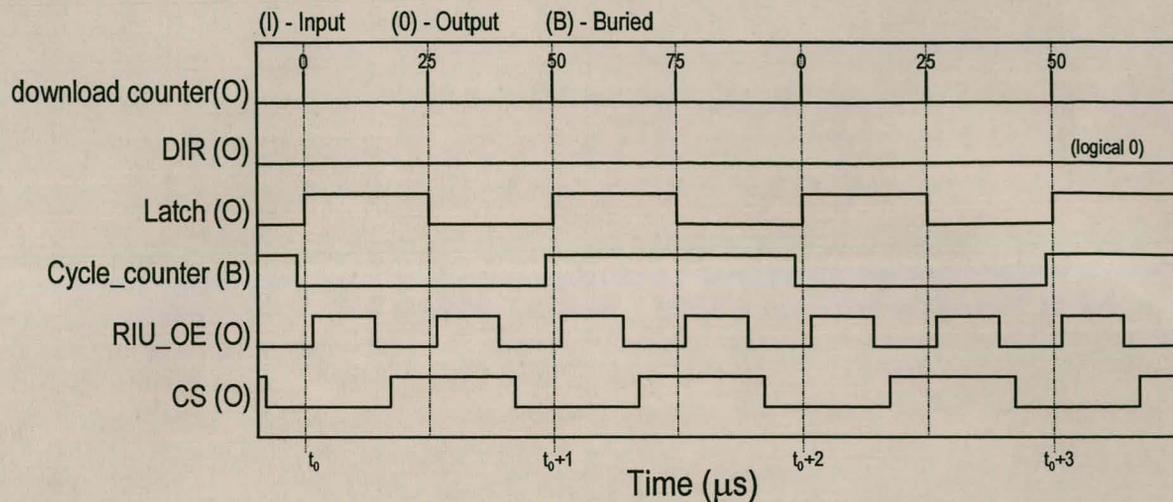


Figure 6-8 Timing waveforms for RAM interface unit control logic during *download mode* (both DRAM modules active)

25, and at  $t = t_0+1$  microseconds it is 50, etc. Figure 6-8 shows the case where both of the DRAM modules are active. In the time from  $t = t_0$  to  $t = t_0+2$  microseconds, a 32-bit word is latched into both RIU 0 and RIU 1. The two 32-bit words are then transferred to the RAU data bus as four 16-bit bytes.

At  $t = t_0$  the value of the *download counter* is zero. At this time *cycle\_counter* is low and *Latch\_clk* is pulsed high. This results in two new 32-bit words being latched into the RIUs from the DRAM modules. When the *download counter* reaches a value of 3, *RIU\_OE* is asserted. At this time *CS* and *cycle\_counter* are low. The upper 16-bits from RIU 0 is therefore transferred to the RAU data bus.

*RIU\_OE* is returned low again when the *download counter* reaches a value of 15, and with the next *mclk* cycle (*download counter* is 16) *CS* is pulsed high. RIU 1 is now selected. When the *download counter* reaches a value of 28, *RIU\_OE* is asserted again for 12 *mclk* cycles and the most significant 16 bits in RIU 1 are transferred to the RAU data bus. *CS* is pulsed low one *mclk* cycle after *RIU\_OE* returns low. When *download counter* reaches 25, *Latch\_clk* is pulsed low again. *Cycle\_counter* is toggled high when the counter value is equal to 48.

At  $t = t_0+1$  microseconds the *download counter* reaches 50 and *Latch\_clk* is pulsed high again. *Cycle\_counter* is high at this time and the RIUs make the least significant 16 bits of the 32-bit

words available to be transferred to the RAU data bus. The control signals are switched again as before (as between  $t = t_0$  to  $t = t_0+1$  microseconds). Accordingly, the least significant 16 bit bytes in the two RIUs are multiplexed to the RAU data bus. *Cycle\_counter* remains high until *download counter* reaches 98. At  $t = t_0+2$  microseconds the *download counter* is reset to zero. The next two 32-bit words are loaded into the RIUs with *Latch\_clk* and the download cycle is repeated.

### 6.3.5 RAM controller and Address generator control logic

In this section the *RAM controller* and *address generator control logic* blocks shown in Figure 6-3 are discussed. During *acquisition mode* and *download mode* the timing parameters of these two logic blocks are dependent upon each other, and for this reason they are discussed together. A block diagram of the control logic is given in Figure 6-9.

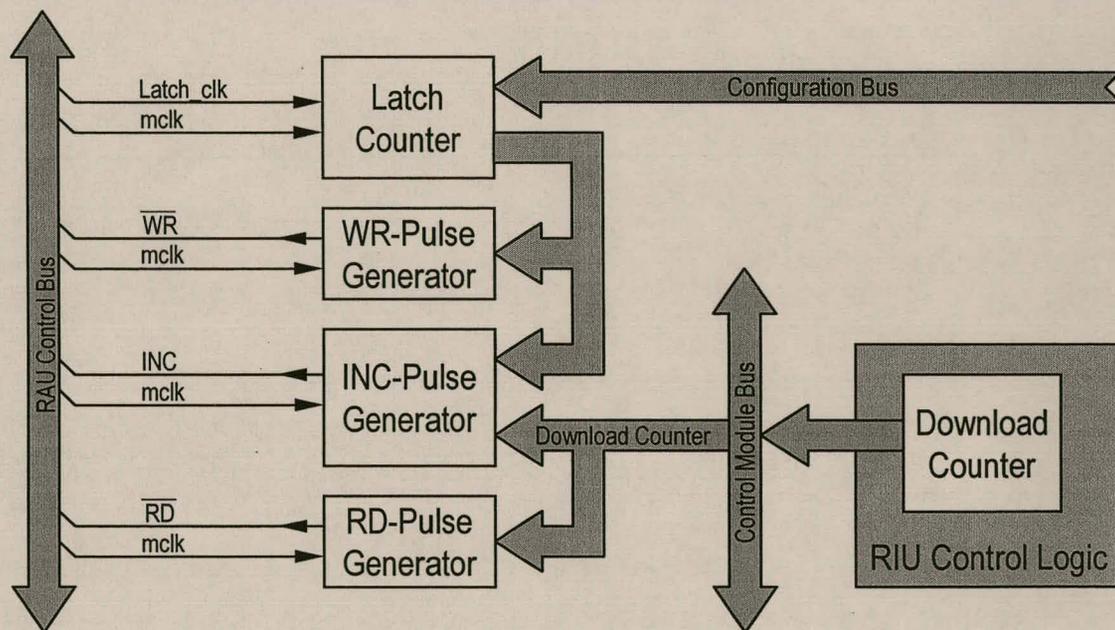


Figure 6-9 RAM controller and address generator control logic in the RAU control module

During *acquisition mode* the *latch counter* monitors *Latch\_clk* and determines when a write (*WR'*) pulse to the RAM controllers is required. In such a case the *latch counter* commands the *WR-pulse generator* to assert *WR'* for one *mclk* cycle. The *latch counter* also determines when the DRAM address in the address generators has to be incremented. The *INC-pulse generator* is then instructed to pulse *INC* high for one *mclk* cycle.

The *RD-pulse* and *INC-pulse generators* are used during *download mode* to control the RD' and INC outputs of the control module. The *download counter* that was discussed in section 6.3.4 determines when RD- and INC-pulses are required. The following subsections discuss the operation of the RAM controller and address generator control logic with reference to *acquisition mode* and *download mode*.

### 6.3.5.1 Acquisition mode

During *acquisition mode* the *latch counter* in Figure 6-9 monitors Latch\_clk to determine how many data bytes have been latched into the active RIUs. If only one DRAM module is active, a WR-pulse is requested after the active RIU has latched in two 16-bit bytes from the RAU data bus. If both RAM modules are active the *latch counter* waits until four 16-bit bytes have been latched into the RIUs.

The *latch counter* then commands the *INC-pulse generator* to assert INC for one mclk cycle. With the following mclk cycle the *WR-pulse generator* is instructed to assert WR'. Accordingly the current DRAM address is incremented, where after the 32-bit data words in the RIUs are written into the selected address location in memory

### 6.3.5.2 Download mode

The same *download counter* that was discussed in section 6.3.4.2 is used for controlling the *RAM controller* and *address generator control logic* during *download mode*. Figure 6-10 shows the timing waveforms involved when data is read from memory. The control signals for the RIU control logic (see Figure 6-8) are also shown to illustrate the synchronisation between the various components.

Latch\_clk is pulsed high at the time that the *download counter* is zero. This causes the 32-bit data presently at the outputs of the DRAM modules to be latched into the two RIUs. When the download counter reaches a value of 3 the INC signal is asserted for one mclk cycle, and the DRAM addresses are incremented to the next memory position. With the next mclk cycle (*download counter* is 4), RD' is asserted and the data at the new memory location is made available to the RIUs. Figure 6-10 shows that when *download counter* reaches zero again, this data is latched into the RIUs and the process is repeated.

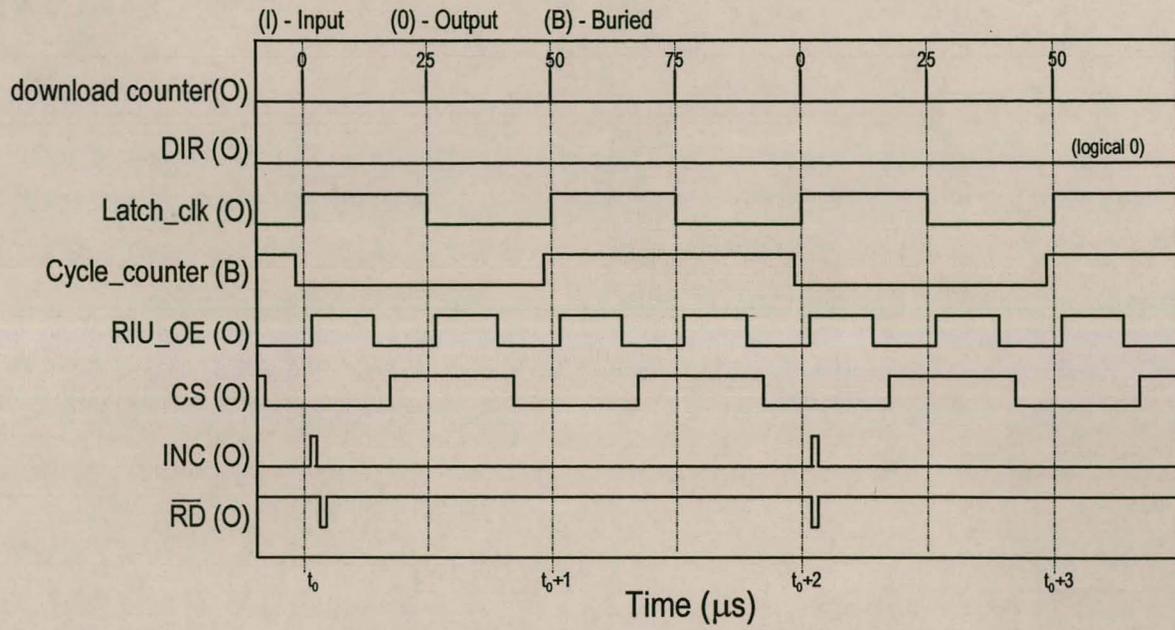


Figure 6-10 Timing waveforms for the DRAM controller and address generator control logic during *download mode* (two RAM modules active)

## Chapter 7

### Interface to host computer

#### 7.1 Introduction

The data acquisition system is controlled from a personal computer (PC) that would typically be situated in a substation environment. A software program has been developed to provide an environment from where the user can control and monitor the data acquisition hardware. The host interface module in the remote acquisition unit (RAU) provides the interface between the acquisition hardware and the host PC. The RAU hardware components are configured via the host interface module. The captured data in the SMM is also downloaded via this module to the host for diagnostic and evaluation purposes. Section 7.2 discusses the hardware implementation of the host interface module. The related software is discussed in section 7.3.

#### 7.2 Host interface module

The RAU host interface module enables the user to control all the data acquisition hardware from the host computer. Configuration of the RAU, as well as initialisation and termination of acquisition cycles is done via this module. Once data has been captured in the SMM of the system, the data can be downloaded to the host PC via the host interface module. A block diagram representation of the interface hardware is shown in Figure 7-1.

As mentioned in chapter 2 both the serial and parallel ports of the host PC interface to the acquisition system. The serial port is used for the transfer of configuration and control data to and from the RAU. A microcontroller is implemented for communication with the serial port of the host PC. The captured acquisition data is downloaded to the host via the parallel port, configured in standard parallel port (SPP) mode. A logic analyser manufactured by *Jobmatch* is used to interface the RAU data bus to the PC.

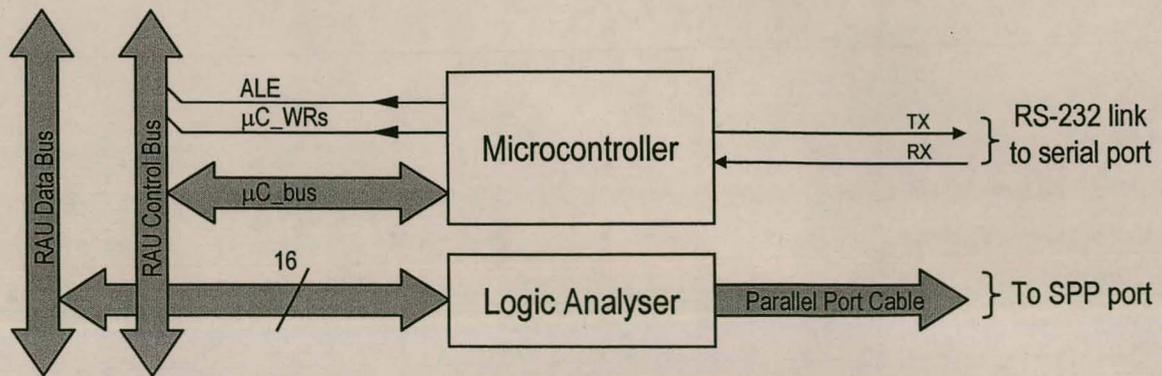


Figure 7-1 RAU host interface module

### 7.2.1 Microcontroller

The microcontroller provides an RS-232 interface between the host computer and the RAU. The user can control the operation as well as operating specifications of the data acquisition system via this link. The devices in the RAU that require configuration data from the host are the RAU control module and the DRAM address generators.

#### 7.2.1.1 Functional description of the microcontroller interface

The control signals providing the interface between the microcontroller and the RAU devices are shown in Figure 7-2.

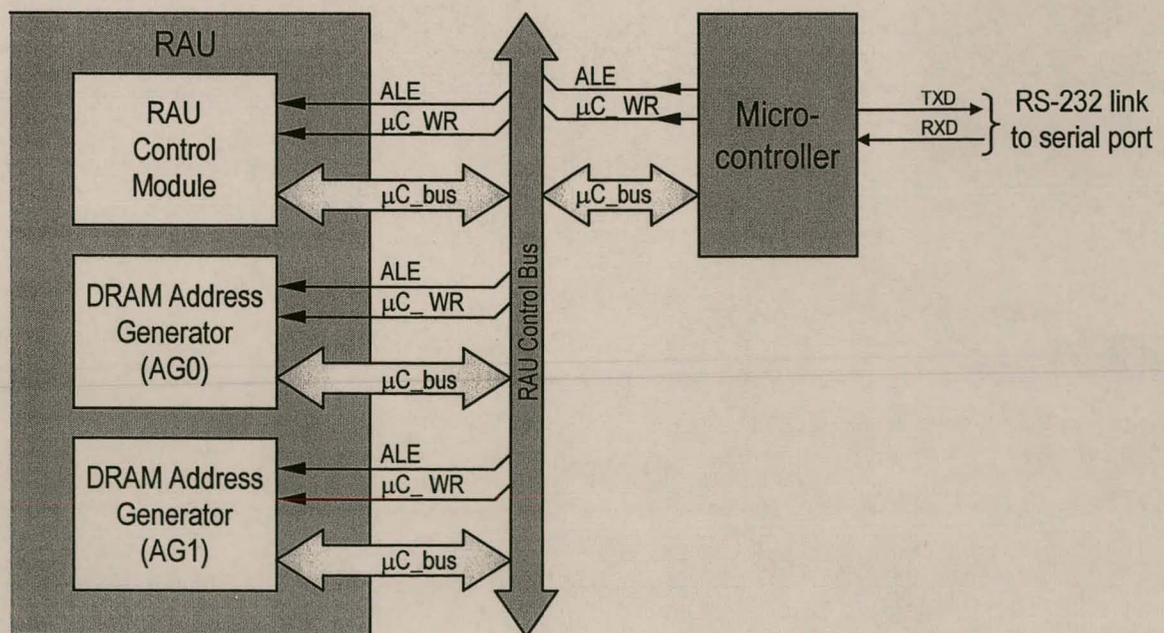


Figure 7-2 Microcontroller interface to the RAU system components

With the data and control signals shown above, the microcontroller can access the various configuration and control registers in the RAU control module and the DRAM address generators. The accessible registers in the RAU control module and address generators were discussed in chapter 6 and chapter 5 respectively. The control module contains three registers that can be configured with the control signals in Figure 7-2, i.e. the *access control register*, and two *configuration registers*. The address generators also each contain an *access control register* and one *configuration register*.

The *configuration registers* in the RAU devices each have a unique register number as discussed in chapters 5 and 6. These register numbers are used to distinguish between the four registers during configuration of the system. In order to change the 8-bit configuration data in one of the configuration registers, the register's number is first transferred to  $\mu\text{C\_bus}$ , and ALE is pulsed high. This causes the register number to be latched into the *access control registers* of the control module and the two address generators. The new configuration byte is then transferred to  $\mu\text{C\_bus}$  and  $\mu\text{C\_WR}$  is pulsed high. On the positive edge of  $\mu\text{C\_WR}$ , the RAU devices determine whether their *access control register* corresponds to the register number of one of its configuration registers. If this is true, the data on  $\mu\text{C\_bus}$  is latched into the corresponding configuration register on the positive edge of  $\mu\text{C\_WR}$ .

### 7.2.1.2 Implementation

The *Atmel* AT89C2051 microcontroller is implemented in the host interface module. The AT89C2051 operates from an internal 8-bit CPU, which executes the microcontroller program from 2 Kbytes of internal flash memory. The device contains 128 bytes of direct addressable internal RAM that is used as data memory. Also included are a number of special function registers (SFRs), a full duplex *universal asynchronous receiver/transmitter* (UART) for serial communications and 15 programmable I/O lines.

### 7.2.1.3 Microcontroller serial port architecture

The UART in the AT89C2051 is utilised for RS-232 communication between the interface module and the host PC. Data is only received from the computer serial port via the RX input of the AT89C2051. The UART can function as an 8-bit UART, 9-bit UART or shift register,

and operates in conjunction with the special function registers (SFRs) SBUF and SCON. All received bytes are stored in a *receive register* that can be accessed by reading from SBUF. SCON is the serial port control register. This register determines the serial port mode, i.e. 8-bit UART, 9-bit UART or shift register. In Appendix G further details are given on the various port modes and the related baud rates of each mode. The serial port interrupt bits namely *transmit interrupt* (TI) and *receive interrupt* (RI) are contained in SCON. TI is set when a data byte is being transmitted (through TXD), and must be cleared by software. When data has been received (through RXD), RI is set and must then also be cleared by software.

#### 7.2.1.4 Programming the microcontroller

The implementation of the microprocessor on schematic level is given in Appendix C with the schematic diagrams of the RAU. A simple protocol has been defined between the control software on the PC and the microcontroller. With every command that the microcontroller must execute, two bytes are sent from the PC. The first byte, referred to as a control byte, contains information on whether the command involves the ALE or the  $\mu\text{C\_WR}$  control signals. The second byte contains the data to be transferred to  $\mu\text{C\_bus}$  and is referred to as a *data byte*. A relatively simple control program is therefore required to manage the operation of the microcontroller. The MCS-51 code for the microcontroller is given in Appendix G. Figure 7-3 shows a flow chart representation of the program.

Initialisation of the microcontroller includes specification of the signals that interface to the RAU, i.e. ALE, WR and  $\mu\text{C\_bus}$ . Serial port interrupts are enabled and the serial port mode and baud rate in the SCON SFR are selected. The UART is configured as an 8-bit UART operating at a baud rate of 9 600. The control program makes use of the B register and the general-purpose flag GF0 (see Appendix G) in the microcontroller. During initialisation both the B register and GF0 are initialised to zero.

When a data byte is received at the serial port through RX, RI in SCON is set by hardware and the received byte is written into SBUF. As was discussed earlier in this section, a *control byte* and *data byte* are alternatively sent from the PC as a pair. The general-purpose flag GF0 is used to keep track of when a received byte is a *control byte* and when it is a *data byte*. GF0 is toggled between '0' and '1' with every received byte. If a byte is received when GF0 is '0', it is

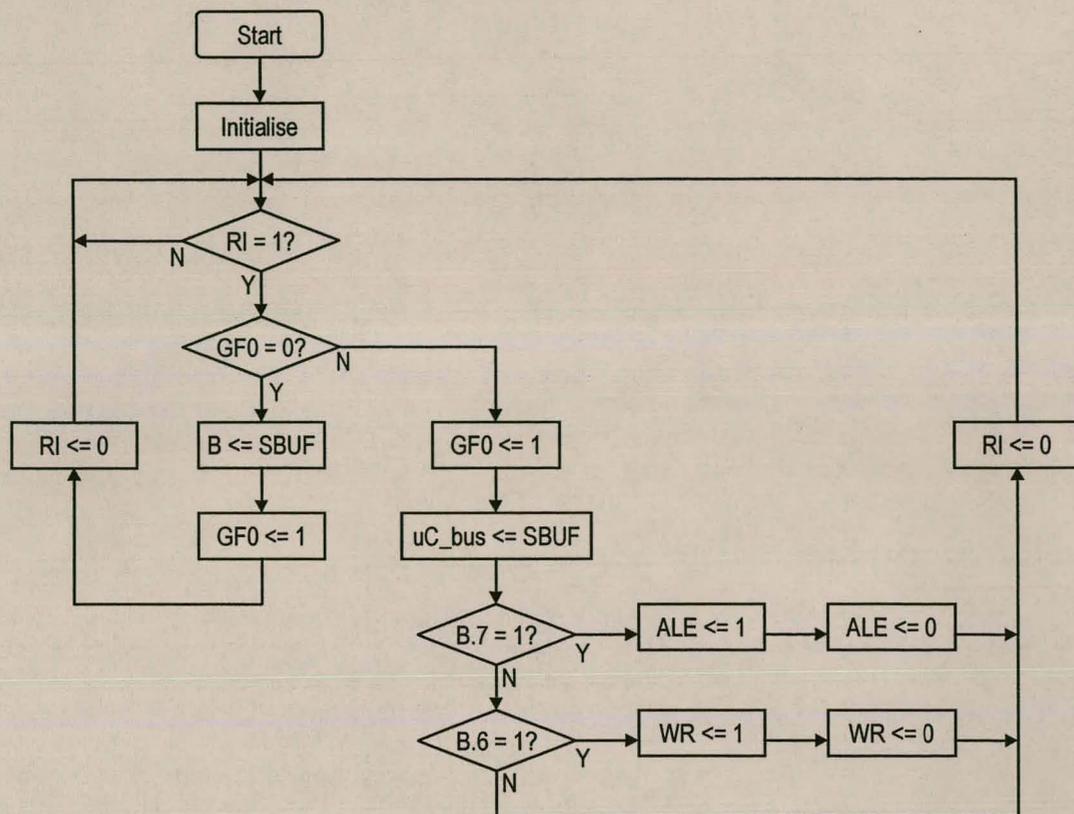


Figure 7-3 Flow chart of the AT89C2051 microcontroller program

a *control byte*, whereas a *data byte* is received when GF0 is '1'.

When a *control byte* is received it is written from SBUF into the B register of the AT89C2051, where after RI is reset. Each *control byte* contains information on whether ALE or  $\mu\text{C\_WR}$  is to be pulsed when the next *data byte* is received from the host. When the microcontroller then receives a *data byte*, it is written onto  $\mu\text{C\_bus}$  and the B register is checked to see whether ALE or  $\mu\text{C\_WR}$  should be pulsed. If the most significant bit of B (B.7) is "1", then ALE is pulsed high and low again. The  $\mu\text{C\_WR}$  signal is pulsed high and low if B.6 is '1'. Finally, RI is reset.

### 7.2.2 Jobmatch logic analyser

A PC based logic analyser, manufactured by *Jobmatch*, is used for downloading the data from the RAU memory to the host PC. This specific logic analyser can sample up to 16 channels at a maximum rate of 200 MHz, with 128 kSamples per channel. It transfers the sampled data to

the PC via the SPP port where it can be saved to disk or viewed (in logic level format) in the *Jobmatch* software environment.

During *download mode* the acquired data samples are transferred from the SMM to the RAU data bus at a rate of 4 samples per 2  $\mu\text{s}$  (see chapter 6). The 16 channels of the logic analyser are connected to the RAU data bus and the data lines are sampled at a rate of 5 MHz. The user can trigger a data capture cycle (128 kSamples) from the PC in the *Jobmatch* environment and save the captured data to disk in text format. A small MATLAB program has been developed which can load the raw data from the text file and plot it on an amplitude/time scale. The MATLAB program is given in Appendix I.

It is important to note that in this project the SMM data is downloaded to the host PC simply to obtain an indication of the validity of the captured data. The simple interface that the *Jobmatch* logic analyser provides between the RAU data bus and the host PC makes it an attractive tool for use in the developed system. The logic analyser can, however, only store 128 kSamples per channel, while the SMM can contain anything between 4 and 64 Mbytes of memory. A small part of the data in the SMM can therefore be transferred to the PC at a time. Furthermore, the logic analyser obtains information on the captured data by sampling the RAU data bus during download mode. The RAU data bus is, however, only valid within certain intervals during download mode. When testing the system, it was found that in spite of this fact the captured data could be validated with the use of the logic analyser.

The constraints specific to the download method describe above could be overcome with the development of a dedicated download module. However, for the purpose of this project a sufficient representation of the captured data could be obtained with the logic analyser. The possibility of a dedicated module is therefore discussed as future work in chapter 9.

### 7.3 Software environment

The data acquisition system is controlled from the host computer in a user-friendly software environment. The software, which is developed in Delphi 3.0, allows the user to specify the operating mode and operating specifications of the data acquisition system from a personal computer. The required software also includes the *Jobmatch* software, which is used to download the captured data to the host PC. During *download mode* the control program and

*Jobmatch* software are run in parallel. Section 7.3.1 discusses the details on the control program while the *Jobmatch* software is discussed in section 7.3.2.

### 7.3.1 System control program

The first part of this section discusses the Delphi control program in terms of the user interface. The second part is dedicated to the lower level interface between the computer and the microcontroller on the RAU.

#### 7.3.1.1 User interface

The system control program allows the user to select the operating mode and operating specifications of the data acquisition system. It then sets these parameters in the RAU control module via the microcontroller. Figure 7-4 shows the form with which the user is presented.

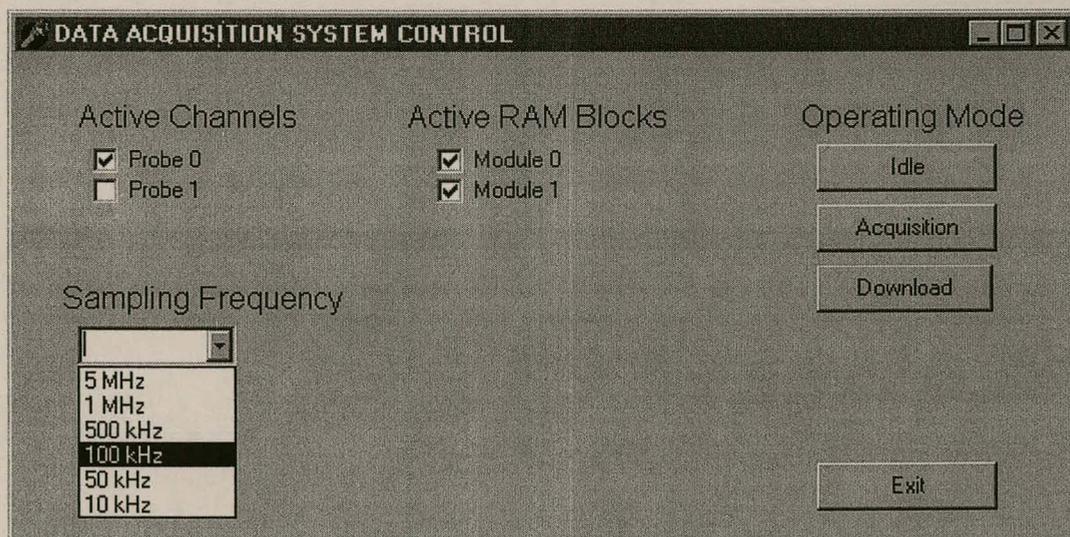


Figure 7-4 User interface for controlling the data acquisition system

The operating mode of the system can be selected by clicking on one of the buttons on the right in the figure above, i.e. Idle, Acquisition, Download or Exit. The Acquisition button sets the data acquisition system in *acquisition mode*. Before this button can be selected, the user must select the active isolated probes and active RAM blocks on the screen as shown in Figure 7-4. The sampling frequency for the system must also be selected from one of 6 frequencies, ranging from 10 kHz to 5 MHz. If any of the parameters have not been specified when the Acquisition button is selected, a warning appears on-screen as in Figure 7-5.

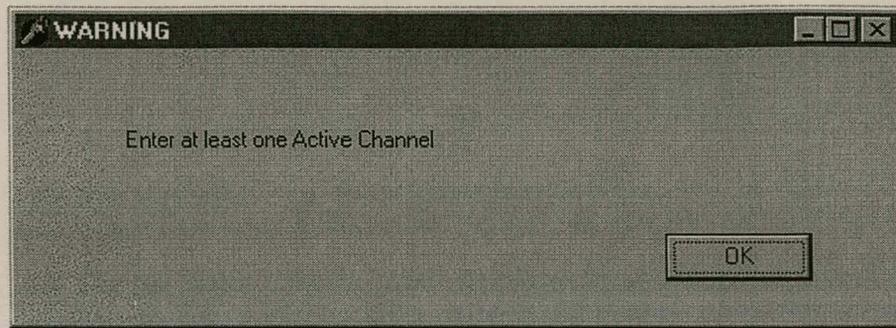


Figure 7-5 Screen warning if all system parameters have not been set

Data acquisition is terminated as soon as one of the other operating mode buttons is clicked. If the user clicks on the Idle or Exit buttons, the system switches to *idle mode*. The Exit button also causes the termination of the control program. The data acquisition system switches to *download mode* when the Download button is selected. Note that this button can only be selected if one of the RAM blocks (module 0, module 1, or both) has been selected. If not, a similar warning as in Figure 7-5 appears on-screen.

### 7.3.1.2 Execution of control program on hardware level

Appendix I gives the programming code for the Delphi 3.0 unit that executes the control commands from the host PC. These commands are sent to the host interface module via the computer serial port. A component namely *TcommPortDriver* [31] is imported in the control program for RS-232 communications via the PC serial port. This component enables full access to the serial port in a *Windows 95* environment and it includes the following features [31]:

- Invisible at run time.
- Can handle first four COM ports (COM1/2/3/4).
- Supports all standard baud rates from 110 to 115200.
- Adjustable COM settings (baud rate, byte size, stop bits, parity, handshaking)
- Sends data by calling *SendData( )* or *SendString( )* methods.

The baud rate of the port is set at 9600 for compatibility with the microcontroller in the host interface module. Furthermore, a byte size of 8-bits is selected with one start and one stop bit.

As discussed in section 7.2.1.1 there are several registers in the RAU control module that

determine the operation of the data acquisition system. These registers are configured from the host PC via the microcontroller. There are also registers in the RAM address generators that require configuration from the host PC. The various configuration registers and the device numbers of the associated RAU modules are summarised in Table 7-1.

Table 7-1 Data acquisition system registers to be configured from the host computer

<i>Op_mode Register</i> (register number 00001010) in RAU Control Module (see section 6.3.1.1)	
Bit 0	'1' – System is in <i>acquisition mode</i> .
Bit 1	'1' – If <i>op_mode.0</i> is '0', the system is in <i>download mode</i> .
Bit 2-7	–
<i>Op_specs Register</i> (register number 00001111) in RAU Control Module (see section 6.3.1.1)	
Bit 0	Sampling frequency bit 0.
Bit 1	Sampling frequency bit 1.
Bit 2	Sampling frequency bit 2.
Bit 3	–
Bit 4	'0' – RAM module 1 is not active. '1' – RAM module 1 is active.
Bit 5	'0' – RAM module 0 is not active. '1' – RAM module 0 is active.
Bit 6	'0' – Isolated probe 1 is not active. '1' – Isolate probe 1 is active.
Bit 7	'0' – Isolated probe 0 is not active. '1' – Isolate probe 0 is active.
<i>Configuration Register</i> (register number 10100000) in Address Generator 0 (see section 5.5.1)	
Bit 0	Reset.
Bit 1	Enable incrementing of DRAM address in module 0.
Bit 2-7	–
<i>Configuration Register</i> (register number 10101111) in Address Generator 1 (see section 5.5.1)	
Bit 0	Reset.
Bit 1	Enable incrementing of DRAM address in module 1.
Bit 2-7	–

The operating mode of the system is set in the *op\_mode* register in the RAU control module, while the operating specifications are set in the *op\_specs* register. The operating specifications include the number of active probes and DRAM modules, as well as the sampling frequency for *acquisition mode*. In the address generators the *configuration registers* can be configured to reset the DRAM addresses, or to enable address incrementing.

The protocol defined between the microcontroller and the host PC was discussed earlier in this chapter in section 7.2.1.4. For every command that the microcontroller must execute, two bytes are sent from the PC. The first byte is referred to as a *control byte*, while the second is referred to as a *data byte* (see section 7.2.1.4). In the following subsections the *control* and *data bytes* for configuring the registers in the RAU control module and address generators are discussed with reference to the form shown in Figure 7-4.

### 7.3.1.2.1 System configuration for *idle mode*

When the user clicks on the *Idle* button in Figure 7-4, the operating mode of the RAU control module is set to *idle mode* in the *op\_mode* register. The address generators are reset by setting the least significant bit in the *configuration registers* of the devices. With the protocol defined between the PC and the microcontroller, the *control* and *data bytes* in Table 7-2 are sent to the microcontroller in the following order.

Table 7-2 Configuration bytes sent to the microcontroller for *idle mode* configuration

8-BIT BYTE	CONTROL /DATA	INTERPRETATION OF BYTES BY RAU DEVICES AND MICROCONTROLLER.	RESULT
10000000	Control	ALE must be pulsed with next received byte.	Set operating mode of system to <i>idle mode</i> .
00001010	Data	This data byte is written into <i>access control registers</i> of RAU control module and both address generators. <i>Op_mode</i> register in RAU control module is selected with 00001010.	
00000000	Control	Pulse WR with next received byte.	
00000000	Data	<i>Op_mode</i> register is set to 00000000. Effectively, system switches to <i>idle mode</i> .	
10000000	Control	Pulse ALE with next received byte.	Reset address generator 0.
10100000	Data	Written into <i>access control registers</i> of RAU control module and address generators. <i>Configuration register</i> in address generator 0 is selected with 10100000.	
00000000	Control	Pulse WR with next received byte.	
00000001	Data	Configuration register set to 00000001. Effectively, address generator 0 is reset.	
10000000	Control	Pulse ALE with next received byte.	Reset address generator 1.
10101111	Data	Written into <i>access control registers</i> of RAU control module and address generators. <i>Configuration register</i> in address generator 1 is selected with 10101111.	
00000000	Control	Pulse WR with next received byte.	
00000001	Data	Configuration register set to 00000001. Effectively, address generator 1 is reset.	

The microcontroller receives the *control* and *data bytes* and drives the 8-bit data bus  $\mu C_{bus}$  and the control signals ALE and  $\mu C_{WR}$  accordingly.

### 7.3.1.2.2 Acquisition mode

If the user has selected to configure the system for *acquisition mode*, the sequence of bits in Table 7-2 is first sent to the host interface module. This ensures that all the system hardware components are initially reset. Table 7-3 gives the *control* and *data bytes* sent to the microcontroller after the reset sequence.

Table 7-3 Configuration bytes sent to the microcontroller for *acquisition mode* configuration

8-BIT BYTE	CONTROL /DATA	INTERPRETATION OF BYTES BY RAU DEVICES AND MICROCONTROLLER.	RESULT
10000000	Control	Pulse ALE with next received byte.	Enable incrementing of address to DRAM 0 by AG0.
10100000	Data	Written into <i>access control registers</i> of RAU control module and address generators. <i>Configuration register</i> in address generator 0 is selected with 10100000.	
00000000	Control	Pulse WR with next received byte.	
00000010	Data	<i>Configuration register</i> set to 00000010. The incrementing of the DRAM address from AG0 is enabled.	
10000000	Control	Pulse ALE with next received byte.	Enable incrementing of address to DRAM 1 by AG1.
10101111	Data	Written into <i>access control registers</i> of RAU control module and address generators. <i>Configuration register</i> in address generator 1 is selected with 10101111.	
00000000	Control	Pulse WR with next received byte.	
00000010	Data	<i>Configuration register</i> set to 00000010. The incrementing of the DRAM address from AG1 is enabled.	
10000000	Control	ALE must be pulsed with the next received byte.	Set operating specifications as selected by user.
00001111	Data	This data byte is written into <i>access control registers</i> of RAU control module and both address generators. <i>Op_specs</i> register in RAU control module is selected with 00001111.	
00000000	Control	Pulse WR with next received byte.	
11110111	Data	<i>Op_specs</i> register is set to 11110111. In this case, both isolated probes and DRAM modules are activated and the sampling frequency is set at 5 MHz.	
10000000	Control	ALE must be pulsed with the next received byte.	Operating mode set as <i>acquisition mode</i>

Table 7-3 (continued)

8-BIT BYTE	CONTROL /DATA	INTERPRETATION OF BYTES BY RAU DEVICES AND MICROCONTROLLER.	RESULT
00001010	Data	This data byte is written into <i>access control registers</i> of RAU control module and both address generators. <i>Op_mode</i> register in RAU control module is selected with 00001010.	
00000000	Control	Pulse WR with next received byte.	
00000001	Data	<i>Op_mode</i> register is set to 00000001. Effectively, system switches to <i>acquisition mode</i> .	

The address generators supply the 23-bit addresses to the DRAM modules during *acquisition mode*. The address outputs of these devices can only be changed if the second least significant bit in the internal *configuration registers* is set. The first 8 bytes in Table 7-3 set this bit in the *configuration registers* of both address generators. If only one DRAM module has been activated by the user, the address incrementing of only that module is enabled as shown above.

The operating specifications of the system as selected by the user are programmed into the *op\_specs* register of the RAU control module. These include the number of active probes and active DRAM modules, and the sampling frequency of the ADCs on the isolated probes. Table 6-2 and Table 6-3 in chapter 6 show the interpretation of the *op\_specs* register by the RAU control module. Finally, the operating mode of the system is set to *acquisition mode* in the *op\_mode* register of the RAU control module.

### 7.3.1.2.3 Download mode

The RAU control module and address generators are automatically reset with the sequence of bytes in Table 7-2, before being configured for *download mode*. Once these *control* and *data bytes* have been transmitted from the host computer, the bytes in Table 7-4 are transmitted.

The sequence of bytes shown below enable the incrementing of the DRAM addresses in both DRAM modules by the respective address generators. Table 7-4 shows the control and data bytes required for enabling both address generators. If the user, however, activated only one of the DRAM modules, the address incrementing in the address generator of only that module

Table 7-4 Configuration bytes sent to the microcontroller for *download mode* configuration

8-BIT BYTE	CONTROL /DATA	INTERPRETATION OF BYTES BY RAU DEVICES AND MICROCONTROLLER.	RESULT
10000000	Control	Pulse ALE with next received byte.	Enable incrementing of address to DRAM 0 by AG0.
10100000	Data	Written into <i>access control registers</i> of RAU control module and address generators. <i>Configuration register</i> in address generator 0 is selected with 10100000.	
00000000	Control	Pulse WR with next received byte.	
00000010	Data	<i>Configuration register</i> set to 00000010. The incrementing of the DRAM address from AG0 is enabled.	
10000000	Control	Pulse ALE with next received byte.	Enable incrementing of address to DRAM 1 by AG1.
10101111	Data	Written into <i>access control registers</i> of RAU control module and address generators. <i>Configuration register</i> in address generator 1 is selected with 10101111.	
00000000	Control	Pulse WR with next received byte.	
00000010	Data	<i>Configuration register</i> set to 00000010. The incrementing of the DRAM address from AG1 is enabled.	
10000000	Control	ALE must be pulsed with the next received byte.	Set system operating mode to <i>download mode</i> .
00001010	Data	This data byte is written into <i>access control registers</i> of RAU control module and both address generators. <i>Op_mode</i> register in RAU control module is selected with 00001010.	
00000000	Control	Pulse WR with next received byte.	
00000010	Data	<i>Op_mode</i> register set to 00000010. Effectively, operating mode set to <i>download mode</i> .	

is enabled. The final four bytes in Table 7-4 effectively set the operating mode of the system to *download mode*.

#### 7.3.1.2.4 Exit

When the user clicks on the Exit button in Figure 7-4, the software program is terminated. The sequence of bytes in Table 7-2 is sent to the microcontroller on the RAU. Accordingly, the system switches to *idle mode*, and further transmission of data from the isolated probes is terminated. The FIFOs/demuxes on the RAU are reset from the RAU control module, and the address generators are reset directly via the microcontroller.

### 7.3.2 Jobmatch software environment

The *Jobmatch* logic analyser discussed in section 7.2.2 samples the data on the RAU data bus during *download mode* and transfers it to the host PC via the SPP port. The software environment from where this device is controlled is illustrated in Figure 7-6.

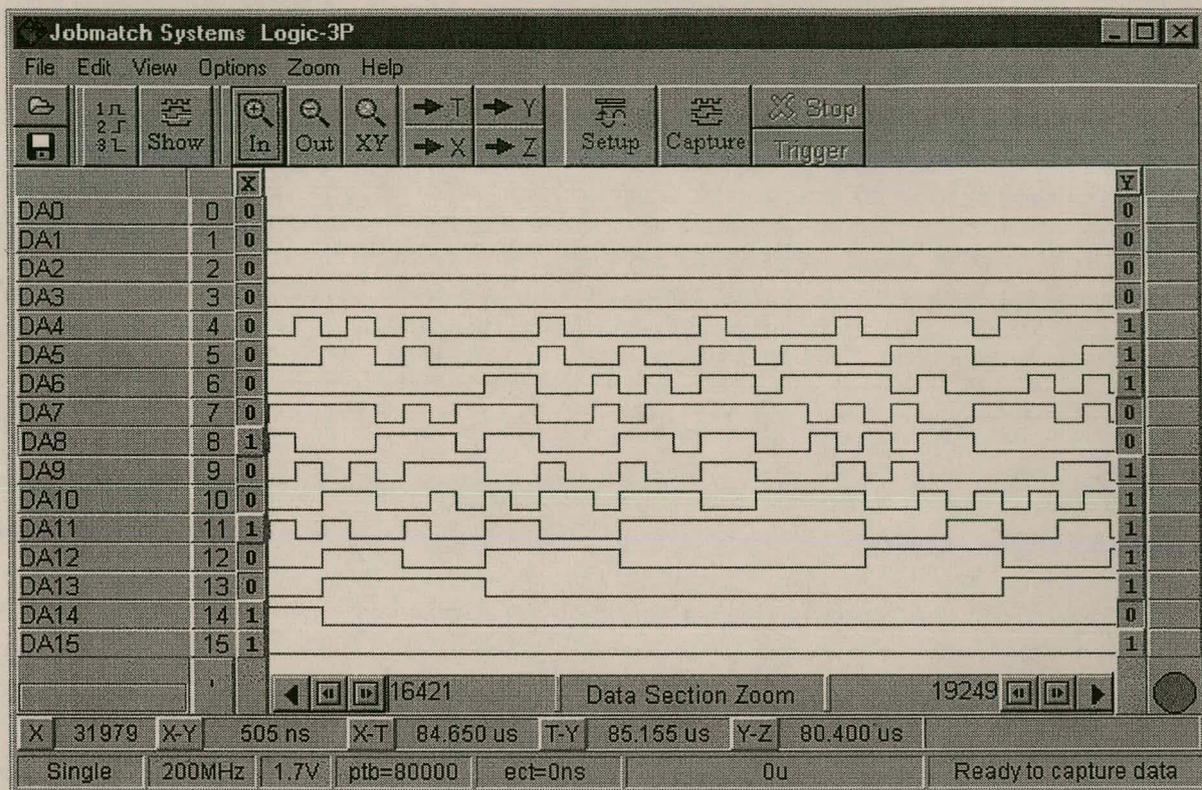


Figure 7-6 *Jobmatch* software environment

The sixteen channels of the logic analyser are connected to the sixteen data lines of the RAU data bus. The sampling frequency of the device is set to 5 MHz [32]. Once the acquisition system has been set to *download mode*, the data in the SMM is repeatedly transferred to the RAU data bus. The logic analyser must then be triggered by the user [32] to sample the data being transferred to the data bus. The logic analyser uses the host for command and display purposes only. Its capturing functions are completely independent from the PC. Up to 131072 data points can be captured at a time.

The captured data points can be saved to disk in hexadecimal text format [32]. Each data sample consists of sixteen bits and is saved as a four digit hexadecimal number. A MATLAB program is given in Appendix I that loads this text file and plots the data on an amplitude/time scale.

## Chapter 8

### Test procedures and results

#### 8.1 Introduction

This chapter discusses the test procedures results obtained in the evaluation of the developed system with reference to the desired system specifications. The approach taken in this chapter involves the evaluation of each system component individually and also the evaluation of the various components in the context of functioning within the system.

Section 8.2 describes the test procedures performed in testing the system. Section 8.3 gives the measurements obtained with these procedures. A window of captured data is studied in section 8.4 in order to determine the validity of the data.

#### 8.2 Test procedures

The data acquisition hardware described in chapters 3, 4, 5, 6 and 7 was tested and evaluated in a laboratory environment. The purpose of the testing procedures was to determine whether the system complies with the specifications as discussed in chapter 2 and to evaluate the performance of the system. The test equipment includes a personal computer, a digital logic analyser and the required software applications and packages. The system was tested with a single isolated probe and DRAM block. Figure 8-1 shows a diagram of the testing environment and setup.

The data acquisition control program that is discussed in section 7.3.1 is run on a personal computer (PC). The PC interfaces to the AT89C2051 microcontroller on the RAU via an RS-232 serial link. The logic analyser discussed in chapter 7 is used for testing the developed hardware. This specific logic analyser can measure up to 16-channels simultaneously at a

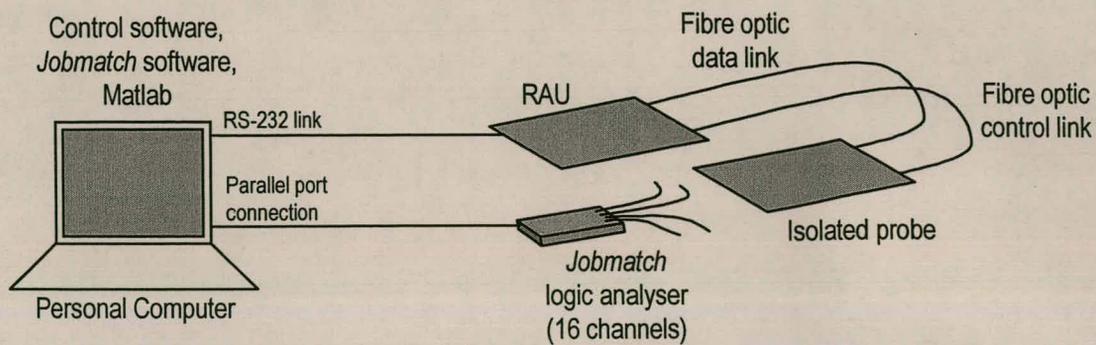


Figure 8-1 Testing environment and equipment

maximum sampling speed of 200 MHz. It connects to the PC parallel port and is controlled via the *Jobmatch* software environment discussed in section 7.3.2.

Testing of the system involved the measuring and capturing of the relevant control and data signals on both the RAU and isolated probe for the various operating modes of the system. The measured signals can be divided into the following categories:

- Control signals output by microcontroller.
- Control/data signals on isolated probe.
- Control/data signals to/from FIFO/demux.
- Control/data signals to/from RIU.
- DRAM controller control signals.
- DRAM address generator control signals.

The captured signals were evaluated in order to determine whether the system operates reliably and according to the specifications. A number of analog input signals were then captured with the acquisition system and the digitised data was manipulated in order to determine the validity of the data.

### 8.3 Measurements involving the performance of the system components

#### 8.3.1 Control signals output by the AT89C2051

The acquisition hardware is controlled via the software control program discussed in section 7.3.1. The AT89C2051 microcontroller provides an RS-232 interface between the host

computer and the RAU, therefore enabling full control of the acquisition system from the PC. A set of consecutive commands is sent from the PC to the RAU with every mode change of the acquisition system. These commands are summarised in Table 7-2, Table 7-3 and Table 7-4. The microcontroller interprets the commands and executes them accordingly. Figure 8-2 shows the signals output by the AT89C2051 when the user selects *idle mode* as the operating mode.

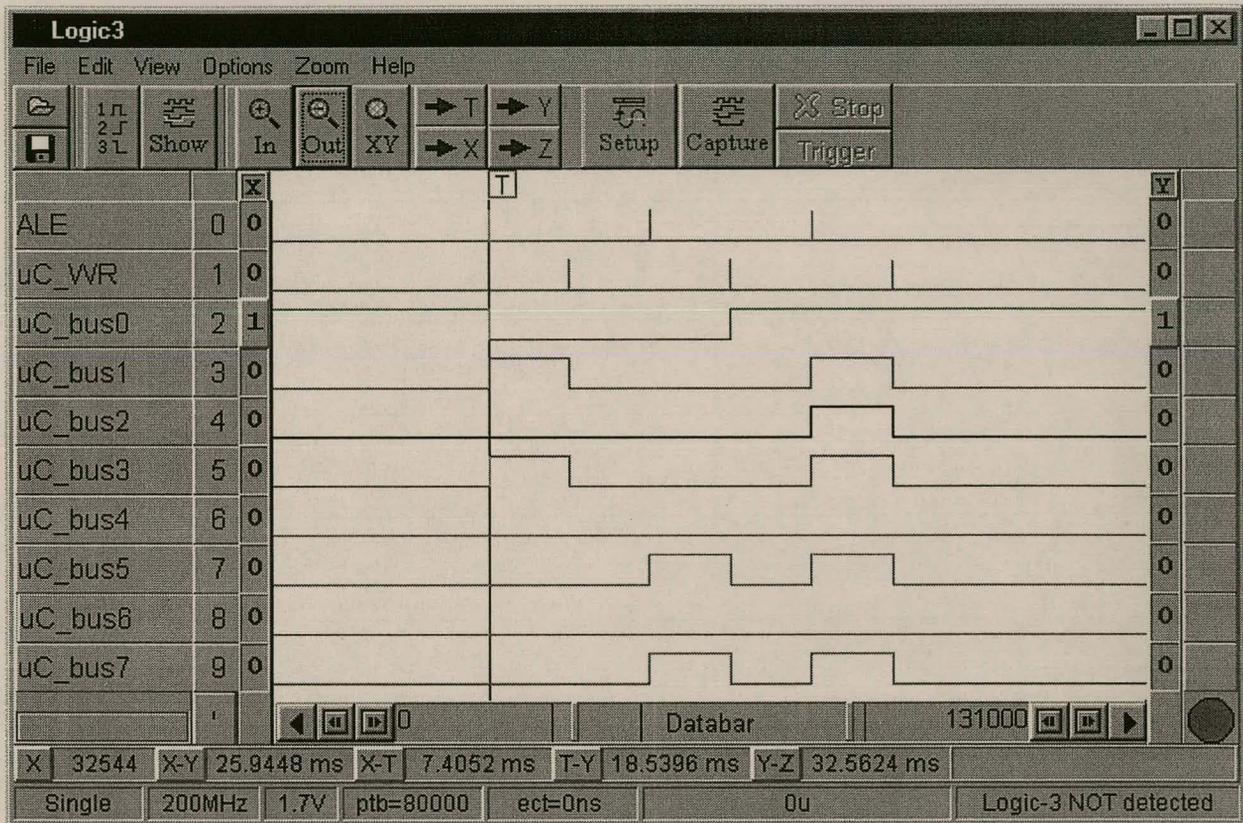


Figure 8-2 AT89C2051 outputs for idle mode configuration

The microcontroller output signals are *address latch enable* (ALE), *write* ( $\mu\text{C\_WR}$ ) and 8 data lines ( $\mu\text{C\_bus}$ ). The first configuration instruction must change the operating mode of the RAU control module to *idle mode*.  $\mu\text{C\_Bus}$  is set to 00001010 and 10  $\mu\text{s}$  later ALE is pulsed high for 1  $\mu\text{s}$ . On the positive edge of ALE the data byte is latched into the *access control registers* of the RAU control module as well as the DRAM address generator. The 00001010 byte provides access to the *op\_mode* register in the RAU control module (refer to Table 7-2). The byte 00000000 is then assigned to  $\mu\text{C\_bus}$  and  $\mu\text{C\_WR}$  is pulsed high for 1  $\mu\text{s}$ . This causes 00000000 to be latched into the *op\_mode* register and the system to function in *idle mode* (refer to Table 7-2).

In addition to changing the operating mode of the RAU control module, the address generators are reset during *idle mode*. This is achieved by setting the *configuration registers* of these devices to 00000001. In order to access the *configuration register* in address generator 0, 10100000 is latched into the *access control registers* of the RAU devices by pulsing ALE. The byte 00000001 is then latched into the *configuration register*, effectively resetting address generator 0. Even though address generator 1 is not implemented the commands to reset it are also sent from the PC.

The control signals (ALE,  $\mu\text{C\_WR}$ ) and data outputs ( $\mu\text{C\_bus}$ ) are controlled similarly as in Figure 8-2 for *acquisition* and *download mode*. The data bytes given in Table 7-3 and Table 7-4 are output in the correct order on  $\mu\text{C\_bus}$ . Ten microseconds after  $\mu\text{C\_bus}$  is valid, either ALE or  $\mu\text{C\_WR}$  is pulsed high for 1  $\mu\text{s}$ . The choice between ALE and  $\mu\text{C\_WR}$  depends on whether the *access control registers* or a specific *configuration register* in the RAU devices is being programmed.

### 8.3.2 Control/data signals on the isolated probe

An isolated probe functions in either *idle mode* or *acquisition mode*, as discussed in chapter 3. The operating mode is determined by the control logic on the probe and depends on whether a sample clock signal is being sent across the fibre optic control link or not. During *idle mode* the RAU control module transmits a logical low across the link, whereas the conversion clock signal for the ADC is transmitted during *acquisition mode*. The conversion clock is routed through the control logic to the ADC, and digital data output by the ADC is transmitted back to the RAU via the fibre optic data link. The function of the control logic during *acquisition mode* is therefore as follows:

- Route conversion clock to ADS801 and enable ADS801 outputs.
- Multiplex 12-bit data outputs from ADS801 to CY7B923 as two 8-bit bytes.
- Supply control signals to CY7B923 with correct timing specifications.

Figure 8-3 shows some of the control/data signals on the isolated probe during *acquisition mode*, with an ADC conversion clock (ADC\_Clock) of 5 MHz. The transmission of a single 12-bit sample output by the ADC takes place between X and Y in Figure 8-3. The timing parameters measured during this time are summarised in Table 8-1.

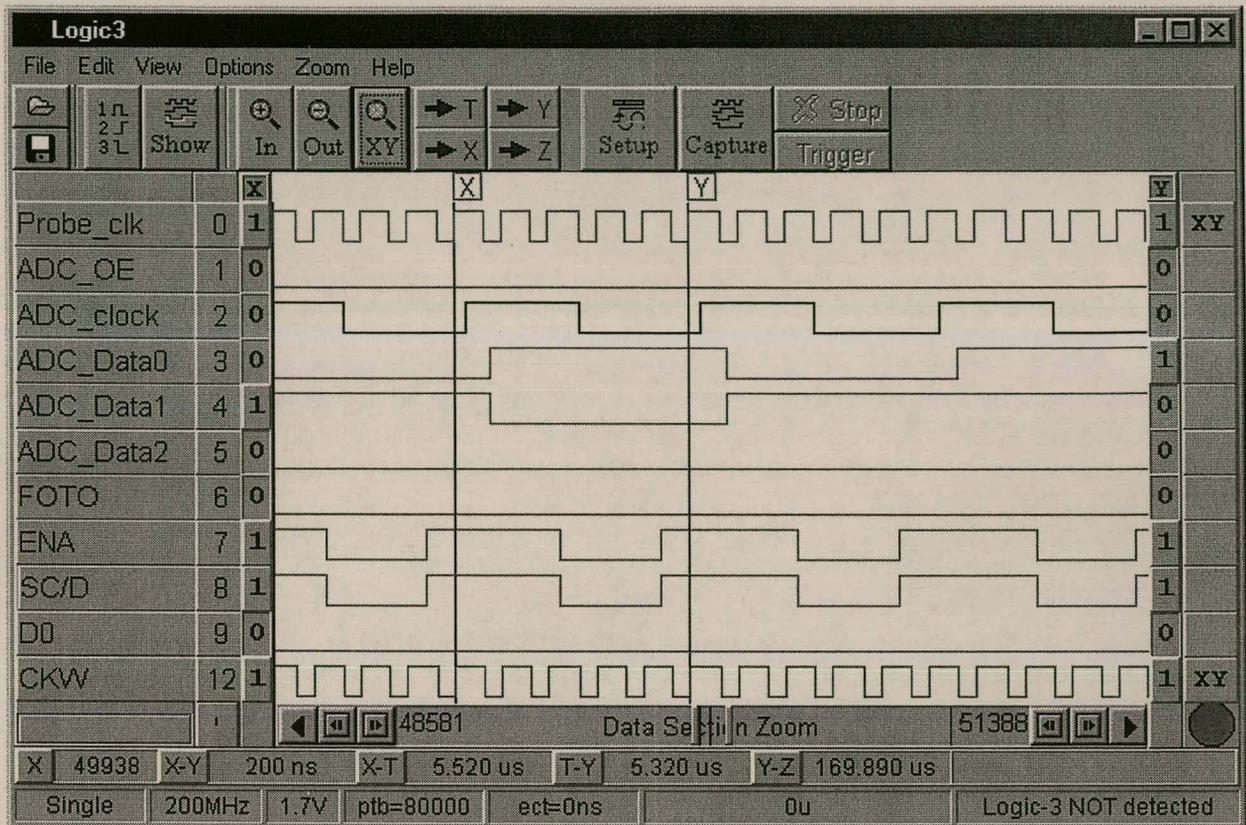


Figure 8-3 Control/data signals measured on the isolated probe during *acquisition mode*

Table 8-1 Measured timing parameters on the isolated probe during *acquisition mode*

Timing interval description	Time
From positive edge of ADC_Clock to ADC_Data valid.	15-20 ns
From positive edge of ADC_Clock to time that control logic module latches in ADC_Data.	2 <sup>nd</sup> positive edge of Probe_clk after ADC_Clock rises
Time it takes 8 MSBs to be valid on D0-D7 after positive edge of Probe_clk.	10ns
Time ENA' takes to pulse low after positive edge of Probe_clk.	10 ns
From when ENA' pulses low until transmission of 8 MSBs.	30 ns
From transmission of 8 MSBs to time that 4 LSBs + 0000 valid on D0-D7.	10 ns
From transmission of 8 MSB to transmission of 4 LSBs + 0000.	40 ns
From transmission of 4 LSBs + 0000 to ENA' high again.	10 ns
Time for transmission of single 12-bit data sample.	200 ns

The control logic module operates with a 25 MHz master clock (Probe\_clk) with 50 % duty cycle, as shown in Figure 8-3. Here it appears that the duty cycle of Probe\_clk deviates from 50 %. This is, however, due to the limited maximum sampling frequency of the logic

analyser (200 MHz).

The 5 MHz conversion clock received via the fibre optic control link is routed to ADC\_Clock and ADC\_Data0 – ADC\_Data11 are enabled by driving ADC\_OE' low. As discussed in section 3.6.2.1, the data on ADC\_Data is latched into the control logic module on the second positive edge of Probe\_clk after ADC\_Clock is high. Figure 8-3 and Table 8-1 show that ADC\_Data is definitely valid at this time.

The same positive edge of Probe\_clk mentioned above causes the eight most significant bits of ADC\_Data to be transferred to the D0-D7 inputs of the CY7B923, and ENA' as well as SC/D' to pulse low. The time it takes D0-D7, ENA' and SC/D' to be set after the positive edge of Probe\_clk was measured as approximately 10 ns. The same 25 MHz signal used as master clock for the control logic is routed to the CKW input of the CY7B23. Therefore, with the next positive edge of Probe\_clk, which is 30 ns later, the data on D0-D7 is transmitted by the C7B923 across the fibre optic data link. Note that FOTO is deactivated during *acquisition mode*. This same clock edge is used to transfer the 4 least significant bits of ADC\_data together with four zeros to D0-D7. The resulting 8-bit byte is then transmitted across the link 30 ns later on the rising edge of Probe\_clk. Finally, ENA' and SC/D' are returned high again.

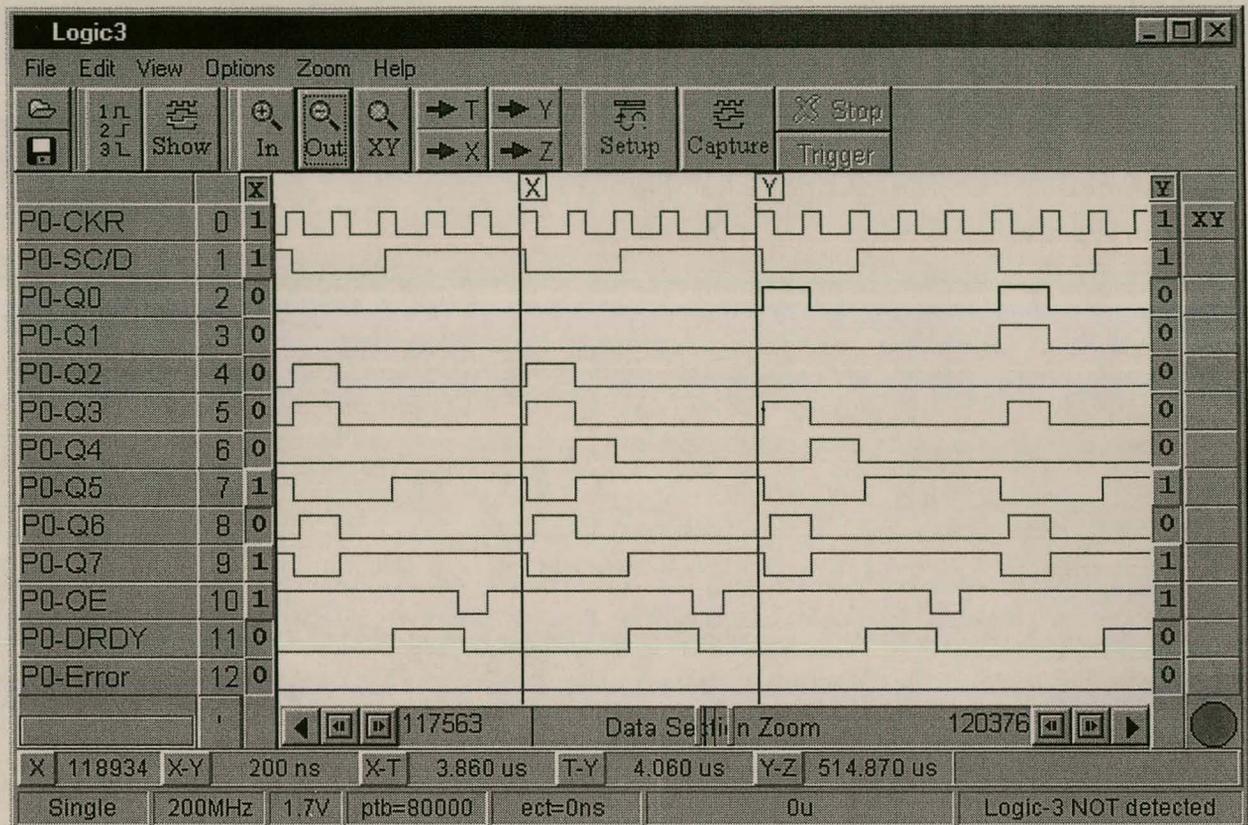
The interaction between the ADC, the control logic and the CY7B923 as shown in Figure 8-3, enables the reliable transmission of up to 16-bits of data, produced at a speed of 5 MHz, across the fibre optic data link in real time.

### 8.3.3 Control/data signals to/from the FIFO/demux

During *idle mode* and *download mode* the reset input (Reset) of the FIFO/demux is permanently asserted. During *acquisition mode* the FIFO/demux must perform the following functions:

- Latch in two 8-bit data bytes from CY7B933.
- Inform RAU control module when it contains a data byte pair.
- React to P0\_OE signal and transfer 16-bit word to RAU data bus.

Various input and output signals of the FIFO/demux were measured during *acquisition mode* and are illustrated in Figure 8-4, with the measured timing parameters given in Table 8-2.

Figure 8-4 Control/data signals to/from the FIFO/demux during *acquisition mode*Table 8-2 Measured timing parameters concerning the FIFO/demux during *acquisition mode*

Timing interval description	Time
From positive edge of P0_CKR until P0_Q0-Q7 changes.	5 – 10 ns
From time that 2 bytes latched into FIFO/demux until P0_DRDY pulsed high.	10 ns
From when P0_DRDY high until PO_OE low.	55 ns
Time that PO_OE is pulsed low.	25 ns
Time from when 16-bit word transferred to RAU data bus until next 8-bit byte latched into device.	70 ns
Time from when first byte latched into device until PO_OE returns high.	130 ns

Two 8-bit bytes are latched into the device and transferred to the RAU data bus in the time between X and Y below. When a data byte is received by the CY7B933, P0\_SC/D' is pulsed low 10 ns after the rising edge of P0\_CKR. At the same time the received data byte is output at P0\_Q0-Q7. With the next rising edge of P0\_CKR, P0\_SC/D' and P0\_Q0-Q7 remain unchanged for at least another 5 ns. This positive edge of P0\_CKR is then used to latch the data on P0\_Q0-Q7 into the FIFO/demux.

In Figure 8-4 it can be seen that the two data bytes constituting a 12-bit data sample from the ADC are received on two consecutive positive edges of P0\_CKR. Once both data bytes have been received, P0\_DRDY is pulsed high. This indicates to the RAU control module that a valid 16-bit word is ready to be transferred to the RAU data bus. The control module responds by asserting P0\_OE, 55 ns later for a period of 25 ns. Once P0\_OE returns high, the FIFO/demux is empty again, and 4 8-bit bytes can be latched into the device again.

### 8.3.4 Control/data signals to/from the RAM interface unit

The RAM interface unit (RIU) provides the interface between the 16-bit RAU data bus and the 32-bit data lines of the DRAM module. When writing to the DRAM this unit performs the following functions:

- Latch in two 16-bit data words from the RAU bus.
- Output the resulting 32-bit word to the DRAM module.

Figure 8-5 and Table 8-3 illustrate the relevant digital control signals to/from the RIU when writing to the DRAM.

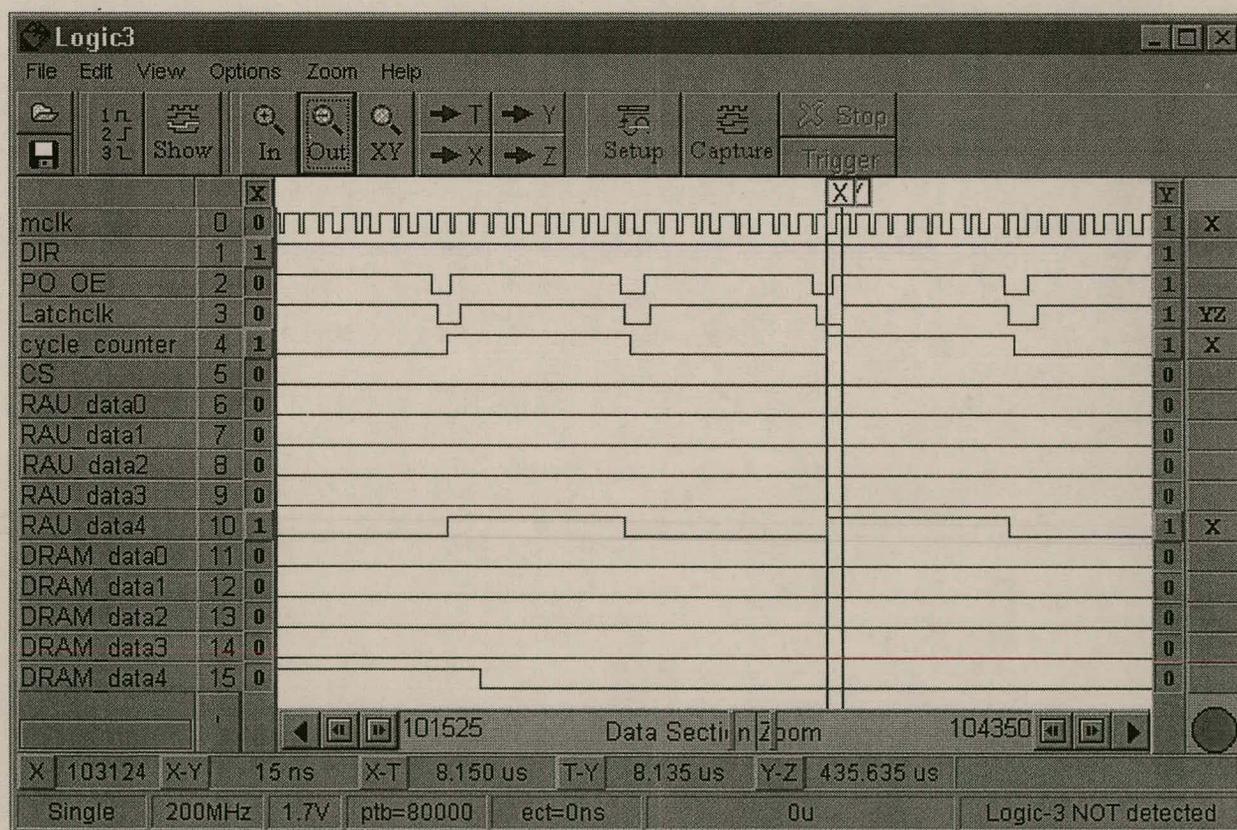


Figure 8-5 Control/data signals to/from the RAM interface unit when writing to DRAM

Table 8-3 Measured timing parameters concerning the RAM interface unit when writing to DRAM

Timing interval description	Time
From time that PO_OE asserted until RAU data valid.	15 ns
Period that PO_OE is asserted.	25 ns
Period that Latch_clk is low.	25 ns
From when cycle_counter is updated until rising edge of Latch_clk.	15 ns
From time that RAU data valid until rising edge of Latch_clk.	15 ns

In Figure 8-5 the RAU control module drives the DIR, Latch\_clk, cycle\_counter and CS inputs of the RIU. The P0\_OE' input of the FIFO/demux is also shown to illustrate certain relevant timing parameters. The 5 least significant bits of the RAU data bus and the DRAM data bus are also shown.

When writing to the DRAM DIR is driven high, indicating to the RIU in which directed data must flow through it. When a valid 16-bit word must be transferred to the RAU data bus, P0\_OE' is asserted for 25 ns by the RAU control module. Approximately 15 ns after P0\_OE' is driven low the data on the RAU data bus is valid. Figure 8-5 shows that at the same time the cycle\_counter input of the RIU is also updated. The level of cycle\_counter determines whether the data is latched into either buffer A or buffer C in the RIU (see section 5.3.1), and must be updated before the rising edge of Latch\_clk.

When the RAU control module returns P0\_OE' high again, it also pulses Latch\_clk high. This rising edge of Latch\_clk causes RAU\_data to be latched into either buffer A or buffer C of the RIU, depending on the level of cycle\_counter. It is clear in Figure 8-5 and Table 8-3 that both RAU\_data and cycle\_counter are valid already 15 ns before the rising edge of Latch\_clk. Therefore the data words latched into the RIU are always valid during *acquisition mode*.

When reading data from the DRAM (during *download mode*) the RIU must perform the following functions:

- Latch in a 32-bit words from the DRAM module.
- Transfer the 32-bit words to the RAU data bus as two 16-bit words.

The measured timing waveforms and parameters concerning the RIU during *download mode* are shown in Figure 8-6 and Table 8-4 respectively.

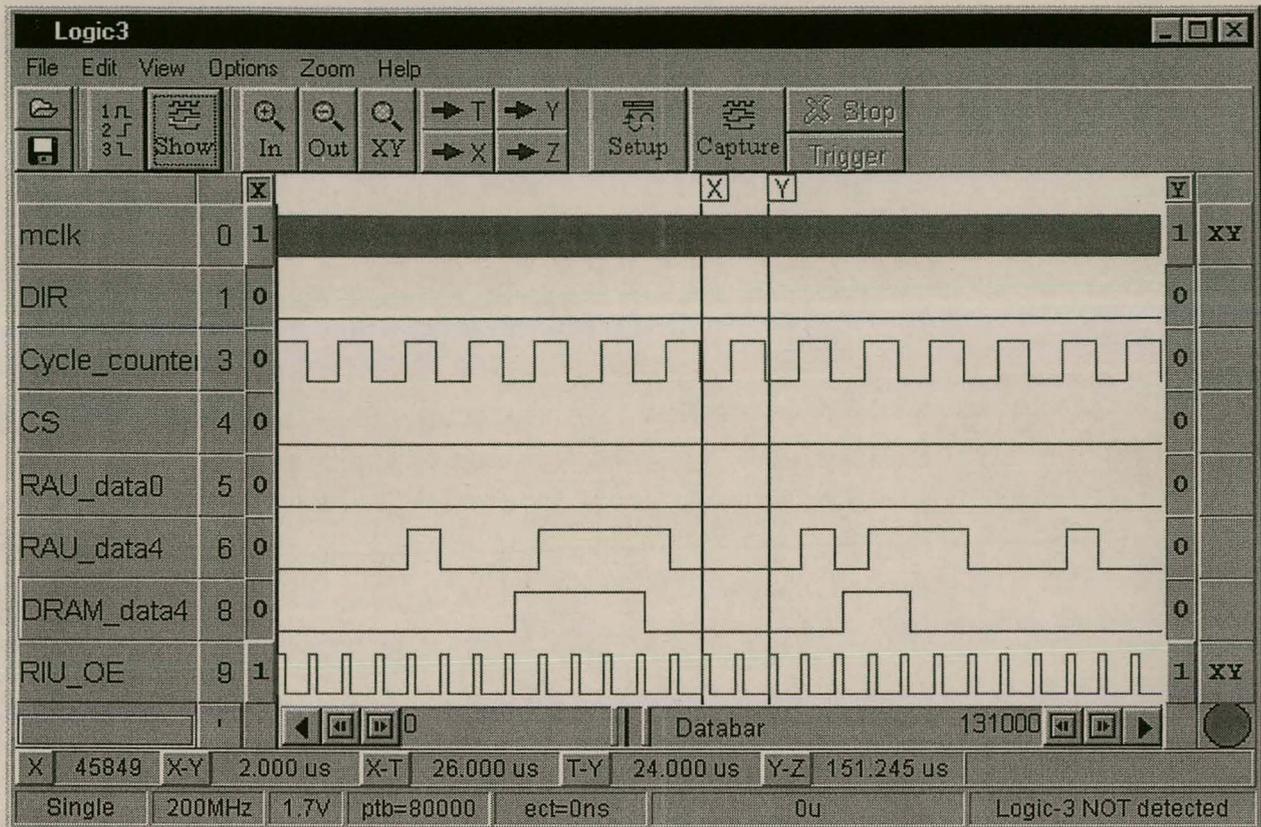


Figure 8-6 Measured control/data signals to/from the RAM interface unit when reading from DRAM

Table 8-4 Measured timing parameters concerning the RAM interface unit when reading from DRAM

Timing interval description	Time
Time that 32-bit data on DRAM_data is valid.	2 $\mu$ s
From when DRAM_data valid until rising edge of Latch_clk.	660 ns
From rising edge of Latch_clk until RAU_data valid.	75 ns
Interval between two consecutive positive edges of Latch_clk.	1 $\mu$ s
From when DRAM_data valid until cycle_counter valid.	620 ns
Time that RIU_OE is asserted.	240 ns
From when RIU_OE asserted until RAU_data valid.	5 ns

During *download mode* data is read from the DRAM module and transferred to the RAU data bus. The DIR input of the RIU is driven low by the control module, causing the RIU to latch in data from the DRAM module. In section 6.3.4.2 it was mentioned that a 32-bit word is latched into the RIU once every 2  $\mu$ s during *download mode* and transferred to the RAU data bus as two 16-bit words. Figure 8-6 and Table 8-4 show that new data is read from the DRAM module every 2  $\mu$ s, as discussed.

Cycle\_counter is driven high by the RAU control module 620 ns after the DRAM data is valid, and 40 ns later Latch\_clk is driven high. This edge of Latch\_clk causes the 32-bit word from the DRAM to be latched into the RIU. The RIU\_OE' input of the RIU is now asserted for 240 ns by the control module, effectively transferring the first 16 bits of the DRAM data to the RAU data bus. RAU\_data becomes valid 75 ns after the rising edge of Latch\_clk. In order to transfer the second 16-bits of the DRAM data to the RAU data bus this process is repeated 1  $\mu$ s later with cycle\_counter low.

### 8.3.5 DRAM controller control signals

The function of the DRAM controller is to manage the read, write and refresh cycles of the DRAM module. Section 5.4 explains that each row in the DRAM has to be refreshed every 15.625  $\mu$ s in order to retain the stored data. This periodic refreshing of the memory is shown in Figure 8-7. The acquisition system is configured for *idle mode* in the figure below.

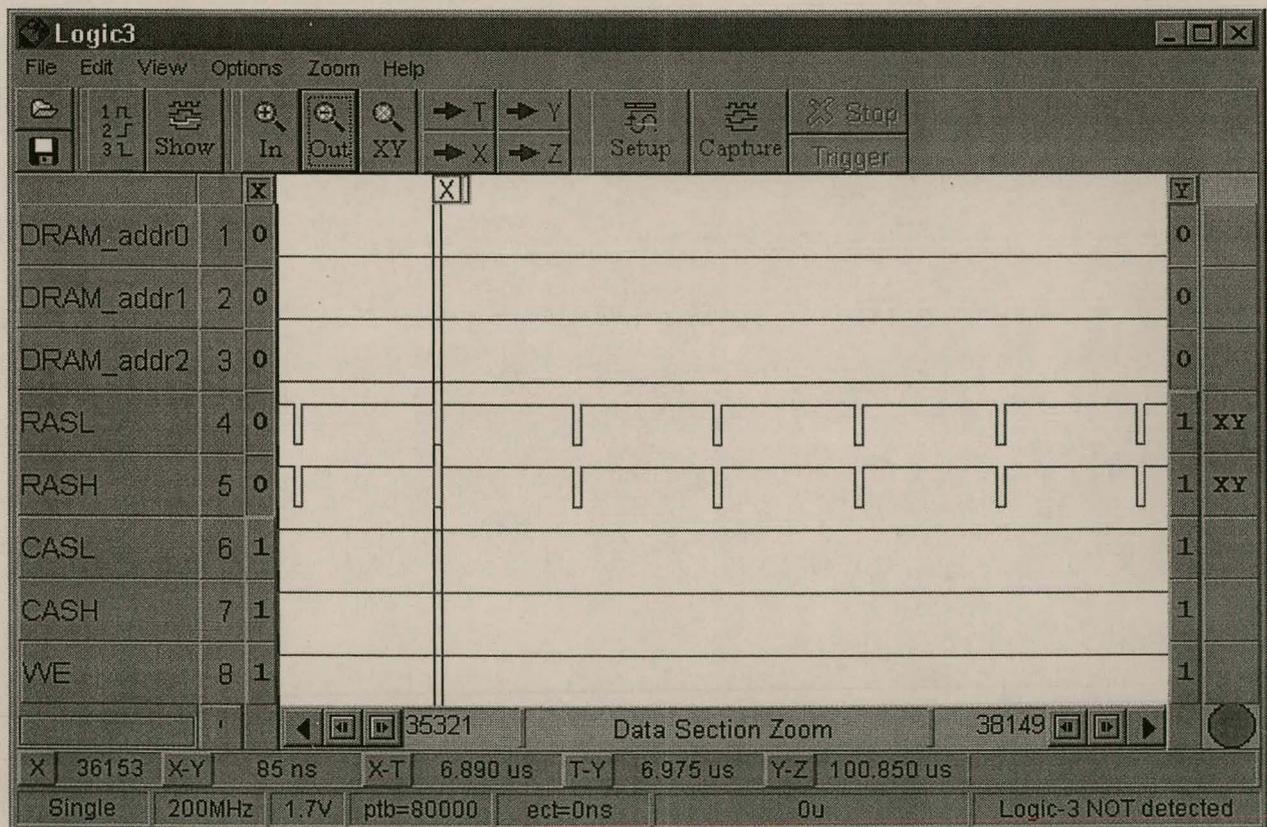


Figure 8-7 Control/data signals concerning DRAM controller during *idle mode*

A row in memory is refreshed with the assertion of the *row address strobe* (RAS') line for the refresh period while the specific row is selected on the memory address lines. Figure 8-7

shows that once every 1.420  $\mu$ s, the RAS'-lines to the DRAM module are asserted for 85 ns. The minimum specifications for these parameters are respectively 15.625  $\mu$ s and 70 ns (see Appendix F). The CAS'-lines remain high during the refresh sequence to conserve power. The 24-bit address provided by the DRAM address generator is ignored during idle mode. The DRAM controller generates the necessary refresh addresses and outputs it to the DRAM module. Once a specific row in memory has been refreshed the SIMM-address is incremented, ready for the next refresh cycle.

During *acquisition mode* and *download mode* the DRAM controller manages the read/write cycles of the DRAM. At the same time it ensures that each row in memory is refreshed within the allowed time period. The control signals measured during *acquisition mode* and *download mode* are shown in Figure 8-8 and Figure 8-9 respectively. Table 8-5 provides further details on the timing of these signals.

The function of the DRAM controller during *acquisition mode* is to supply the necessary control signals to the DRAM for the execution of both write and refresh cycles. When the system operates at full speed, the 32-bit data words to be captured in memory are produced by the RIU at a rate of 2.5 MHz. A data word must therefore be written into memory every 400 ns. Section 5.4.1 discusses that upon a write-cycle request from the RAU control module, the DRAM controller first executes the next refresh cycle. It then proceeds to perform the requested write-cycle.

The WR'-input of the DRAM controller is pulsed low every 400 ns by the RAU control module. This initiates a data cycle write request. The DRAM controller responds by firstly refreshing a row in memory. The refresh row address is output to the DRAM module and RASL' is pulsed low for 80 ns. RASL' returns high and must remain so for the RAS-precharge time which is 70 ns in this case. The controller then proceeds to write the 32-bit data word into memory. As previously mentioned, the 10 least significant bits of DRAM-addr are used as the row address for 4Mbyte SIMMs (see section 5.4.3). These bits are transferred to the SIMM-addr inputs of the memory and RASL' is strobed for 80 ns. The WE' input of the DRAM is pulsed low before the column address is latched into the memory. This results in a write cycle rather than a read cycle. Thereafter the column address is latched into the memory by transferring DRAM-addr10-19 to the address inputs of the SIMMs and strobing CASL' low for 40 ns. WE' is also returned high 30 ns after CASL'.

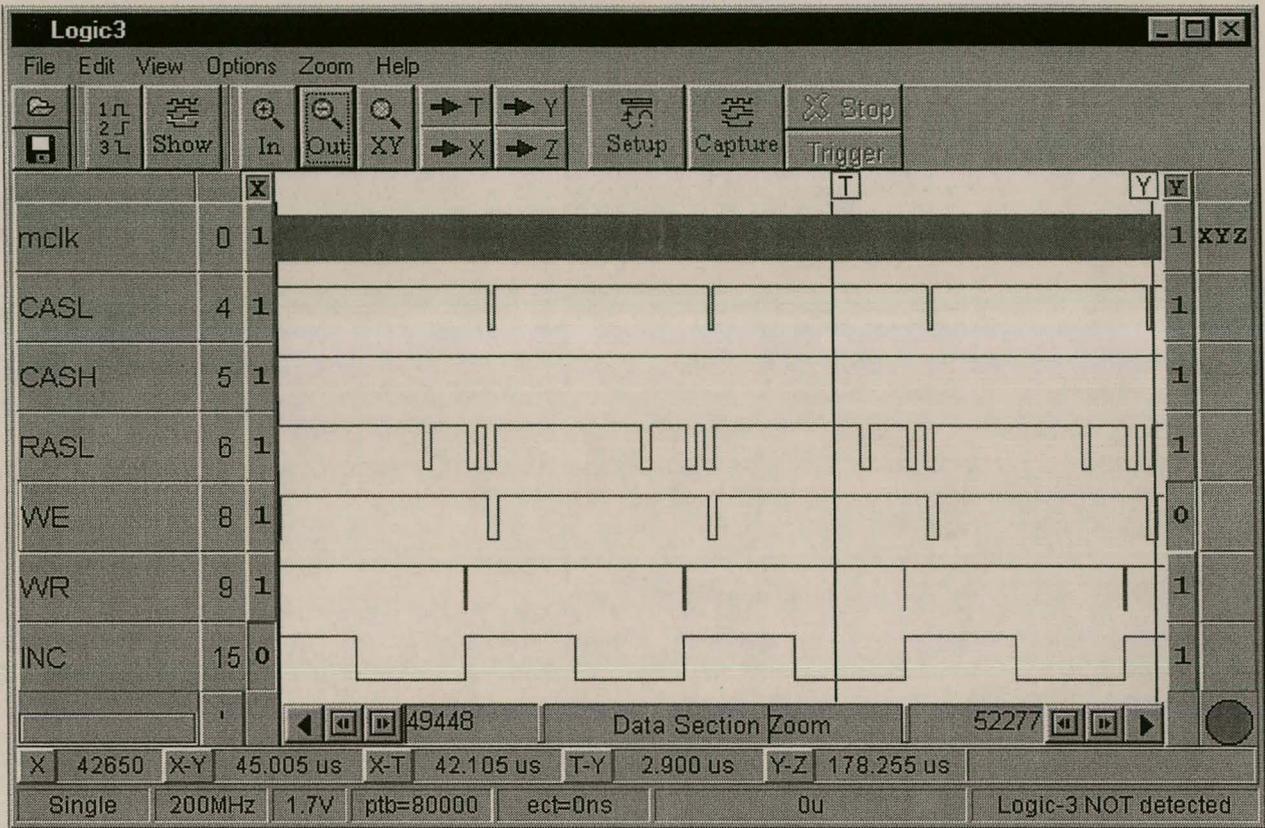


Figure 8-8 DRAM controller signals when writing to memory

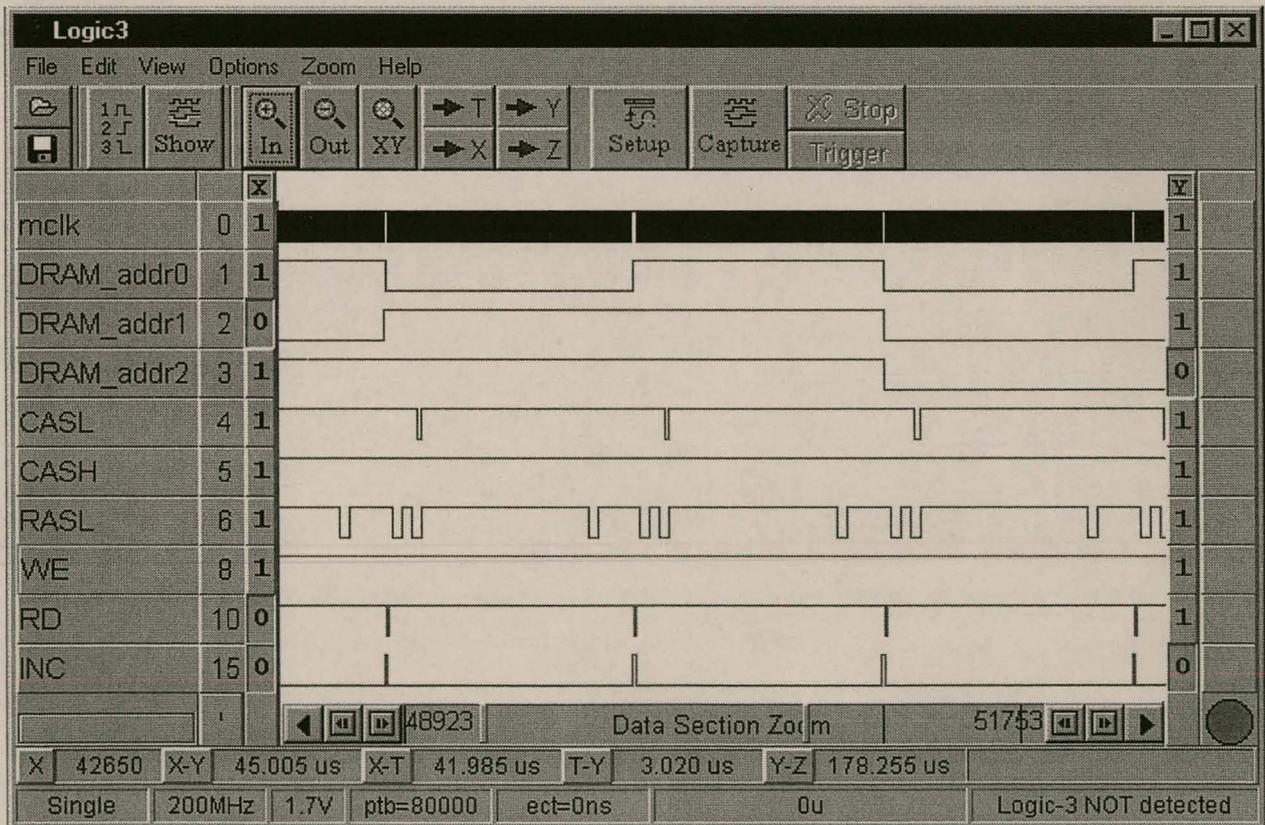


Figure 8-9 DRAM controller signals when read from memory

Table 8-5 Measured timing parameters concerning the DRAM controller during *acquisition mode* and *download mode*

Timing interval description	Specified time	Measured time
RASL'-pulse duration for refresh cycle	70 ns – 10 $\mu$ s	85 ns
Minimum RASL' pre-charge time after refresh	50 ns min	70 ns
Minimum time between two consecutive refresh cycles	15.625 $\mu$ s	400 ns
RASL' pulse duration for data cycle	70 ns – 10 $\mu$ s	85 ns
RASL' low to CASL' low delay time for data cycle	20-52 ns	45 ns
CASL' pulse duration for data cycle	18 ns – 10 $\mu$ s	40 ns
RASL' low to CAS' high for data cycle	70 ns min	85 ns
Minimum CASL' pre-charge time after data cycle	5 ns min	155 ns
WE' pulse duration for write-cycle	10 ns min	90 ns
WE' low before CASL' low setup time for write-cycle	0 ns min	20 ns
WE' low before RASL' high setup time for write-cycle	18ns min	60 ns
WE' low before CASL' high setup time for write-cycle	18ns min	60 ns
WE' high after CASH' high hold time for write-cycle	0 ns min	30 ns
Maximum time between two refreshes during <i>download mode</i>	15.624 $\mu$ s	1.580 $\mu$ s

The data captured in the system memory is transferred back to the RAU data bus during *download mode*. Figure 8-9 shows that once every 2  $\mu$ s a 32-bit word is read from consecutive memory addresses. The RAU control module initiates a read cycle by asserting the RD' input of the DRAM controller. As when writing to the DRAM, the controller reacts by immediately executing the next refresh cycle. The refresh row address is output to the memory and RASL' is pulsed low for 80 ns. In order to read a 32-bit data word from memory, the output signals of the DRAM controller are controlled as for a write-cycle. During a read-cycle WE' is, however, kept high. The row address (DRAM-addr0-9) and column address (DRAM-addr10-19) of the memory position to be read from are output to the DRAM while RASL' and CASL' are respectively strobed. It is also clear from Figure 8-9 that the refreshing of the rows in memory always takes place within the specified 15.625  $\mu$ s.

### 8.3.6 DRAM address generator control signals

The DRAM address generator provides the 23-bit memory address to the DRAM controller. The output address of the address generator is incremented on every positive edge of INC thus

is received from the RAU control module. The DRAM address is incremented during *acquisition mode* just before a data word is to be written into memory. For *download mode* the output address incremented each time a data word is to be read from memory. The address generator is disabled when the system is configured for *idle mode*.

The signals received by the DRAM address generator during *acquisition mode* together with the four least significant bits of the output address are given in Figure 8-10. Various other signals are also shown as an illustration of the timing parameters involved.

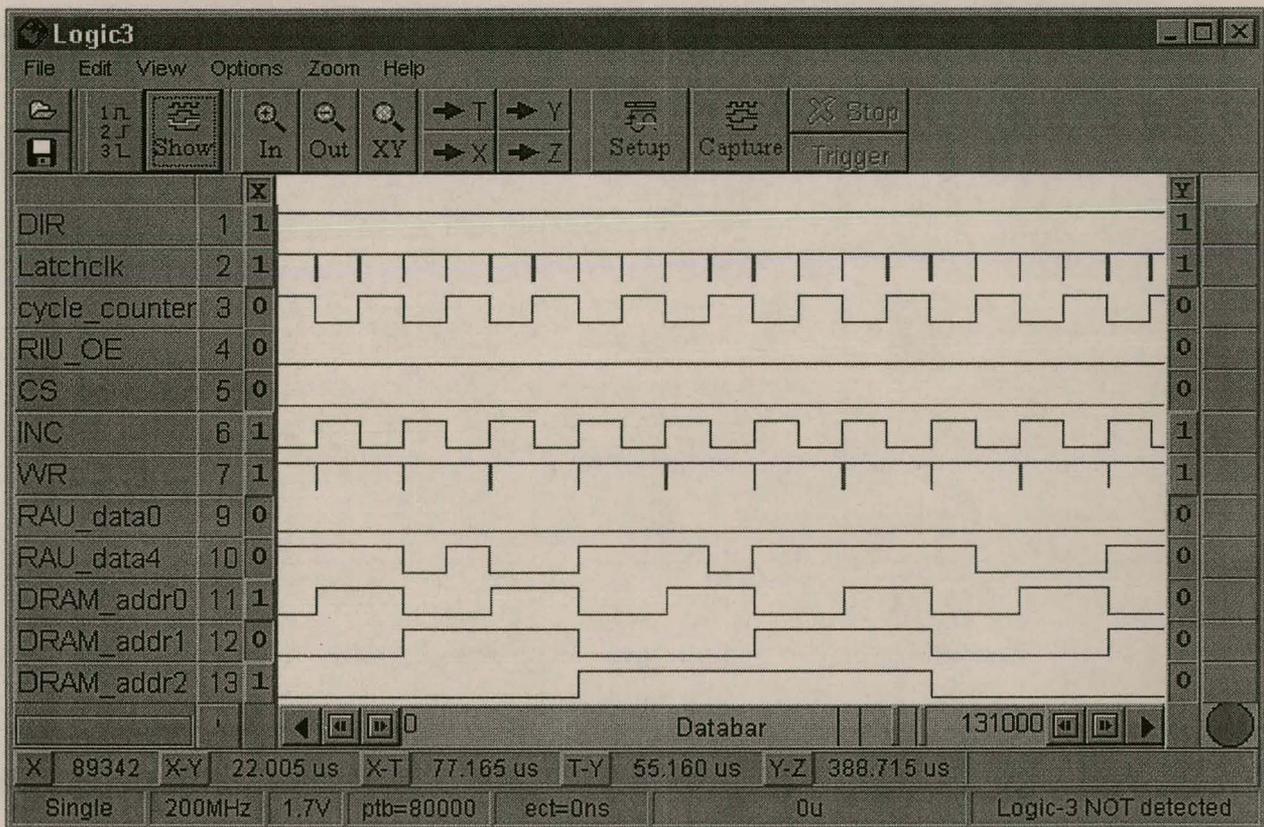


Figure 8-10 Signals concerning DRAM address generator during *acquisition mode*

The DRAM address is incremented when two 16-bit bytes have been latched into the RIU and the data is ready to be written into memory. This situation arises when cycle\_counter is '1' on the rising edge of Latch\_clk. INC is then switched high and the 23-bit DRAM address is incremented and valid 15 ns later. On the next rising edge of Latch\_clk, INC is returned low. WR' is asserted at the same time that INC is pulsed high. The negative edge of WR' indicates to the DRAM controller that data must be written into memory. The DRAM controller performs or completes a refresh cycle before attending to the write-cycle. This gives the address

generator enough time to ensure that the 23-bit address is valid when it is latched into the DRAM controller.

The measured signals concerning the address generator during *download mode* are similar to those shown in Figure 8-10. After two bytes have been transferred from the RIU to the RAU data bus the DRAM address is incremented. RD' is asserted 20 ns after the positive edge of INC. At this stage the new DRAM address is already valid and the data from the relevant memory position is read.

#### 8.4 Validation of the captured data

Several analog input signals were digitised and captured with the data acquisition system. The data was then downloaded to the host PC with the method described in section 7.2.2. The data in memory is transferred to the RAU data bus as 16-bit bytes. A window of the data is sampled on the RAU data bus with the *Jobmatch* logic analyser. The data is then downloaded to a file on the computer and plotted on an amplitude/time scale with the aid of a MATLAB program. Figure 8-11 shows the result of 10 kHz sine wave, which was captured, downloaded and plotted.

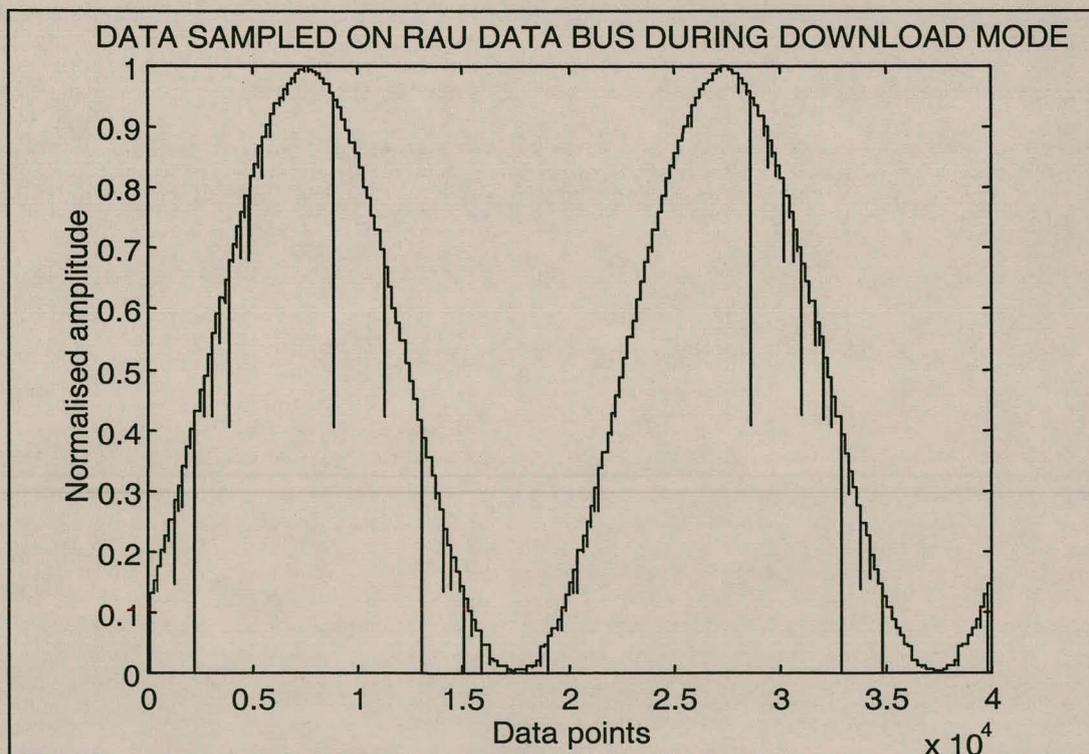


Figure 8-11 Plot of captured data with 10 kHz sine wave analog input

The graph in Figure 8-11 gives an indication that the analog input signal was digitised and captured successfully. The data plotted here was sampled on the RAU data bus during *download mode*. The RAU data bus is, however, valid only during certain time intervals in *download mode* as previously mentioned. This explains the data points in Figure 8-11 that deviate from the sine waveform.

## Chapter 9

### Conclusions and recommendations

#### 9.1 Introduction

This thesis presents the development of a high-speed data acquisition system for high voltage environments. This chapter firstly presents a brief summary of the project in section 9.2. Section 9.3 gives the important conclusions that are made after completion of the project. Further improvements and future work are pointed out in the final section of the chapter.

#### 9.2 Project review

There is a growing interest worldwide in the use of wide band signals in high voltage environments for applications such as substation monitoring and system monitoring in high voltage environments. These wide band signals range from switching and lightning impulses to partial discharge phenomena and corona. Advanced research has been done in recent years on the utilisation of standard substation equipment as wide band transducers. Promising results have been obtained in this field thus far, but a lack of affordable and suitable data acquisition systems is a definite problem.

A prototype system is developed in this thesis that provides a basis for a solution to the problem of expensive and unsuitable data acquisition equipment. A cost-effective system topology is proposed that is optimised for high voltage power system environments. The system is eventually to be integrated on a permanent or extended temporary basis into the test environment. Important aspects considered in establishing the system topology include the ability to monitor transducers spaced physically apart as well as EMC aspects.

The objectives of the project can be summarised as follows:

- Highlighting the important aspects to be considered when measuring wide band signals in HV environments.
- The establishment of a prototype data acquisition system topology to serve as basis for future work in this field, and the illustration of certain design concepts concerning such a topology.
- An evaluation of available technology for utilisation in a wide band data acquisition system, and a demonstration of this technology.

A study was conducted on the possible applications and research fields for a system as developed here. Subsequently, the specifications of the system could be established. The cost of system components, the status of available technology and the implications of the technology for design and manufacture played important roles in determining a suitable system topology. The acquisition system was constructed and finally tested in a laboratory environment.

### 9.3 Conclusions

The developed data acquisition system provides an excellent basis for the design of affordable and suitable systems for capturing wide band signals in high voltage environments. The system complies with the minimum specifications established in chapter 2. The design concepts utilised in the project were shown to be successful. The concepts implemented supply a platform for further development and improvements on the system. The system has, however, not yet been integrated into a high voltage environment. This process will almost certainly provide a number of problems that have not been addressed in this project.

Compensation for EMC and earthing problems is of paramount importance when performing any type of measuring in high voltage environments. The topology of the developed system eliminates much of these problems. Furthermore, it allows for the following features:

- Isolation between sensing hardware and computer architecture components.
- Synchronised sampling of 2 channels.
- Real-time storage of data for up to 2 channels.
- Remote control of acquisition hardware.
- Possible real-time digital signal processing.

In designing a relatively complex system, as in this project, it is advantageous to maintain a measure of modularity between the various components that are implemented in the system. The topology of the developed system is based on a modular type design where each module, other than the control module, is relatively unaware of the operation of the other system modules. As previously mentioned, this thesis highlights the important considerations and concepts involved in acquiring data in high voltage environments, and it supplies a platform for future research in this field. The inherent modularity of the developed acquisition system ensures that future adjustments can be made without having to redesign the complete system.

Extensive use of field programmable gate array (FPGA) technology in the acquisition system increases the programmability and flexibility of the system as a whole. As previously mentioned, real-time digital signal processing features, such as filtering and spectral analysis, can easily be integrated in the acquisition path. Upgrading of some of the programmable logic devices is also relatively flexible in cases where more logic cells are required for DSP operations.

The use of optical fibre in the system brings with it a number of advantages and disadvantages. Both the fibre optic data links and fibre optic control links operate differentially. Communications over these links therefore eliminate the grounding problems that are often encountered in high voltage environments, especially when performing high frequency measurements. In these environments large electric fields are present and the low susceptibility of optic fibre to electro-magnetic interference is a further advantage.

The high-speed (> 5MBd) optical transmitters and receivers that are available today mostly operate at PECL logic levels. Some form of interfacing logic is therefore required in order to integrate the devices with standard TTL devices. There are various dedicated devices on the market that provide a relatively easy interface between PECL optical components and otherwise TTL systems. It was found that both the optical components and the interfacing devices, however, have high power consumption rates. This poses certain problems for battery-operated systems, and there is much room for optimisation in this respect.

An interesting fact that became evident with progression of the project is that the available technology still places limits on the performance achievable with an acquisition system as developed here. The majority of the implemented hardware devices were configured to

perform close to their maximum performance abilities. Such devices include the EPLDs, FPGAs and the on-board memory. In some instances devices are available with better performance characteristics than those used. A balance has to be maintained, however, between system performance and the eventual cost of the system. This thesis presents an acquisition system that complies with the minimum specifications required, and at the same time provides an affordable solution for a wide variety of applications.

#### 9.4 Recommendations

The development of a data acquisition system, as in this thesis, covers only part of the scope of a larger project, which is the design of a complete system for capturing wide band data in high voltage environments. This thesis presents a solution for the synchronised capturing of two wide band analog signals at low voltages ( $< 5V$ ). Suitable transducers and sensors have to be selected or designed and interfaced with the developed system in order to utilise the system in high voltage environments.

A methodology will have to be developed for a *battery-operated* system when utilising the acquisition system in the field as in a substation environment. The power consumption of the system at this stage is of the order of 5 Watt, which is very high for a battery-operated system. The major current-sinking components are the optical transmitters and receivers and the HOTLink™ transmitter/receiver pair. The high clock frequencies that the programmable logic devices operate from also add a substantial amount to the power consumption. Possible improvements in this respect include a study to determine if lower power optical devices and interface devices have not yet entered the market. Furthermore, the use of low voltage logic devices can be considered. This will, however, required more than one supply voltage in the system leading to more complex designs.

The method described in chapter 7 for downloading the captured data from the on-board memory to the host PC is only useful for testing purposes and in a laboratory environment. In the field, the RAU is to be situated in a remote location and controlled via an RS-232 link. The most important aspects to consider when selecting a suitable download link are the following:

- High data transfer rates must be obtainable.
- The data received on the PC side must have high integrity.

Considering the amount of memory to be downloaded, data transfer rates of high in the MBd range will be required. RS-232 and even RS-485 are therefore not suitable. Optical fibre links comply with both the criteria mentioned above, but will again have an effect on the cost of the system as well as the power consumption. A solution is still to be found in this respect.

The use of configurable logic devices for digital signal processing has expanded greatly in the past few years. Technology is now allowing the manufacture of devices with size and speed not previously obtainable with programmable devices. The acquisition system makes extensive use of FPGA technology. This makes it relatively easy to integrate digital processing features, such as filtering and spectral analysis, in the existing path of acquisition. An ideal location for the implementation of such features is in the FPGA that is implemented on the isolated probes.

At the time of design it was decided to implement DRAM SIMMs as memory for the system for the reasons discussed in chapter 5. Dynamic RAM was very cheap relative to static RAM, and if more memory is required a new SIMM is simply plugged in. DRAM SIMMs are, however, linked to the personal computer industry, which continually progresses at a very fast rate. The system memory module will therefore have to be redesigned continually in order to keep up with the evolving computer industry. For this reason it could be advantageous to reconsider the design of the system memory module, and to consider the implementation of a smaller memory bank in the form of SRAM.

## **Acknowledgements**

To Dr. H.J. Vermeulen for his vision and guidance throughout this project.

To the Nikon crew for their support and motivation.

To my husband Sven, for his unfailing support, motivation and patience, and for maintaining a measure of sanity at times.

To my parents, Rulan and Rhalda.

To my fellow students Melanie, Faatima, Johann, Cornel and Gerhard for their encouragement and the help offered when needed.

## References

- [1] C.R. Evert, *Conducted HF Discharge Recognition*, Research Project Proposal Report TRR/E/96/EL137, ESKOM, October 1996
- [2] A.C. Britten and C.R. Evert, *Instrument Transformer Condition Monitoring*, Research Project Proposal Report TRR/E/98/EL029, ESKOM, March 1998
- [3] G.L. van der Zel, *Condition Monitoring of High Voltage Apparatus*, Research Report TRR/E/98/EL019, ESKOM, May 1998
- [4] P.J. de Klerk, *Transformer Failures, Monitoring Techniques & Strategies*, Research Report TRR/E/96/EL137, ESKOM, October 1996
- [5] M. El-Hami, L.L. Lai, D.J. Daruvala, A.T. Johns, *A new travelling-wave based scheme for detection faults on distribution systems*, IEEE transactions on power delivery, Vol. 7, No. 4, October 1992, pp. 1825-1833
- [6] R.K. Aggarwal, J.A.S.B. Jayasinghe, Z.Q. Bo, and A.T. Johns, *A new approach for discriminating between switching and fault generated broadband noise on series compensated EHV transmission lines*, International Conference on Advance in Power System Control, Operation & Management, APSCOM-95, Hong Kong Convention & Exhibition Centre, November 9-11, 1995.
- [7] H.J. Vermeulen, *Wideband modelling and in situ parameter estimation of a capacitive voltage divider with nonlinear damping*, Ph.D Thesis, Stellenbosch University, December 1995
- [8] H.J. Vermeulen, A.C. Britten, M.W. Roberts and J.M. Strauss, *Frequency Response of Capacitive Voltage Transformers and Insulated Current Transformers*, Southern African Universities Power Engineering Conference, Stellenbosch, South Africa, January 1998, pp. 85-88
- [9] *Fibre optic transmitter and receiver data links for 266 MBd*, Hewlett-Packard Co., USA, 1996
- [10] *CY7B923/33 HOTLink Transmitter/Receiver Datasheet*, Cypress Semiconductor Corporation, June 1995
- [11] P. Horowitz and W. Hill, *The Art of Electronics 2<sup>nd</sup> Edition*, Cambridge University Press, Cambridge, 1989
- [12] G. Esterhuizen, *Design of GPS Sampling System*, Final year project, Stellenbosch University, December 1998

- [13] *12-Bit, 25 MHz Sampling Analog-To-Digital Converter*, Burr-Brown Corporation, USA, September 1996
- [14] *Low Cost Fibre-Optic Links for Digital Applications up to 155 MBd*, Application Bulletin 78, Hewlett-Packard Co., USA, 1997
- [15] *HOTLink™ Design Considerations*, Cypress Semiconductor Corporation application note, June 1995
- [16] D.A Nattrass, *Partial discharge detection in high voltage equipment*, Butterworth & Co., 1989
- [17] F. Kreuger, *Partial discharge detection in high voltage equipment*, Butterworth & Co., 1989
- [18] M. Botha, *Modelling of Electromagnetic Interference Generated by Static Var Compensators*, ESKOM report TRR/E/97/EL134, 1997
- [19] H.C. Ferreira and O. Hooijen, *Power line communications: an overview*, SAIEE transactions, September 1995, pp 145-161
- [20] N.P. Tlhatlhetji, *CVTs and insulated CTs for voltage transient measurements in high voltage systems*, ESKOM technology group research report TRR/E/98/EL184, pp 25-30.
- [21] *Digital recorder measurements in high voltage impulse tests Part 1: Requirements for digital recorders*, IEC Publications 1083 – 1, 1991
- [22] H. Ryan, *High voltage engineering and testing*, Peter Peregrinus Ltd, United Kingdom, 1994
- [23] *Data Acquisition Overview*, Quatech: Quality Technology from Application to Solution, [http://www.quatech.com/public/daq\\_over.htm](http://www.quatech.com/public/daq_over.htm), October 1999
- [24] C. Peacock, *Interfacing the PC*, <http://www.senet.com.au/~cpeacock>, October 1999
- [25] *Introduction to the IEEE 1284-1994*, Warp Nine Engineering, <http://www.fapo.com/ieee1284.htm>, July 1999
- [26] *W83977AF Winbond I/O Data Sheet*, Winbond Electronics Corp, April 1998 rev. 0.52
- [27] A. Clements, *Microprocessor system design, 68000 hardware, software and interfacing*, PWS-Kent Publishing Company, Boston, 1992
- [28] *TM497BBK32, TM497BBK32S 4194304 by 32-bit Dynamic RAM Module*, Texas Instruments, Houston, June 1995
- [29] *TM124BBK32F, TM124BBK32U 1048576 by 32-bit Dynamic RAM Module*, Texas Instruments, Houston, December 1994

- [30] K. Heunis and H.J. Vermeulen, *A Distributed Intelligent Data Acquisition System for High Voltage Substation and Test Environments*, 33<sup>rd</sup> Universities Power Engineering Conference (Upec'98), Napier University, Edinburgh, UK, September 1998, pp. 210-213.
- [31] CSE Computer Solution Engineering powered by Borland Delphi, <http://www.deutch.de/ftp.html>, October 1999
- [32] *Logic-3p 200 MHz PC-Based logic analyser User's manual*, Jobmatch systems, April 1998
- [33] *Frequently Asked Questions about HOTLink*, Cypress Semiconductor Corporation, June 1995

## Appendix A

### CY7B923/33 data and control codes

#### A1 Valid data characters

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fg hj	abcdei	fg hj
D0.0	000	00000	100111	0100	011000	1011
D1.0	000	00001	011101	0100	100010	1011
D2.0	000	00010	101101	0100	010010	1011
D3.0	000	00011	110001	1011	110001	0100
D4.0	000	00100	110101	0100	001010	1011
D5.0	000	00101	101001	1011	101001	0100
D6.0	000	00110	011001	1011	011001	0100
D7.0	000	00111	111000	1011	000111	0100
D8.0	000	01000	111001	0100	000110	1011
D9.0	000	01001	100101	1011	100101	0100
D10.0	000	01010	010101	1011	010101	0100
D11.0	000	01011	110100	1011	110100	0100
D12.0	000	01100	001101	1011	001101	0100
D13.0	000	01101	101100	1011	101100	0100
D14.0	000	01110	011100	1011	011100	0100
D15.0	000	01111	010111	0100	101000	1011
D16.0	000	10000	011011	0100	100100	1011
D17.0	000	10001	100011	1011	100011	0100
D18.0	000	10010	010011	1011	010011	0100
D19.0	000	10011	110010	1011	110010	0100
D20.0	000	10100	001011	1011	001011	0100
D21.0	000	10101	101010	1011	101010	0100
D22.0	000	10110	011010	1011	011010	0100
D23.0	000	10111	111010	0100	000101	1011
D24.0	000	11000	110011	0100	001100	1011
D25.0	000	11001	100110	1011	100110	0100
D26.0	000	11010	010110	1011	010110	0100
D27.0	000	11011	110110	0100	001001	1011
D28.0	000	11100	001110	1011	001110	0100
D29.0	000	11101	101110	0100	010001	1011
D30.0	000	11110	011110	0100	100001	1011
D31.0	000	11111	101011	0100	010100	1011

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fg hj	abcdei	fg hj
D0.1	001	00000	100111	1001	011000	1001
D1.1	001	00001	011101	1001	100010	1001
D2.1	001	00010	101101	1001	010010	1001
D3.1	001	00011	110001	1001	110001	1001
D4.1	001	00100	110101	1001	001010	1001
D5.1	001	00101	101001	1001	101001	1001
D6.1	001	00110	011001	1001	011001	1001
D7.1	001	00111	111000	1001	000111	1001
D8.1	001	01000	111001	1001	000110	1001
D9.1	001	01001	100101	1001	100101	1001
D10.1	001	01010	010101	1001	010101	1001
D11.1	001	01011	110100	1001	110100	1001
D12.1	001	01100	001101	1001	001101	1001
D13.1	001	01101	101100	1001	101100	1001
D14.1	001	01110	011100	1001	011100	1001
D15.1	001	01111	010111	1001	101000	1001
D16.1	001	10000	011011	1001	100100	1001
D17.1	001	10001	100011	1001	100011	1001
D18.1	001	10010	010011	1001	010011	1001
D19.1	001	10011	110010	1001	110010	1001
D20.1	001	10100	001011	1001	001011	1001
D21.1	001	10101	101010	1001	101010	1001
D22.1	001	10110	011010	1001	011010	1001
D23.1	001	10111	111010	1001	000101	1001
D24.1	001	11000	110011	1001	001100	1001
D25.1	001	11001	100110	1001	100110	1001
D26.1	001	11010	010110	1001	010110	1001
D27.1	001	11011	110110	1001	001001	1001
D28.1	001	11100	001110	1001	001110	1001
D29.1	001	11101	101110	1001	010001	1001
D30.1	001	11110	011110	1001	100001	1001
D31.1	001	11111	101011	1001	010100	1001
D0.2	010	00000	100111	0101	011000	0101
D1.2	010	00001	011101	0101	100010	0101

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fghj	abcdei	fghj
D2.2	010	00010	101101	0101	010010	0101
D3.2	010	00011	110001	0101	110001	0101
D4.2	010	00100	110101	0101	001010	0101
D5.2	010	00101	101001	0101	101001	0101
D6.2	010	00110	011001	0101	011001	0101
D7.2	010	00111	111000	0101	000111	0101
D8.2	010	01000	111001	0101	000110	0101
D9.2	010	01001	100101	0101	100101	0101
D10.2	010	01010	010101	0101	010101	0101
D11.2	010	01011	110100	0101	110100	0101
D12.2	010	01100	001101	0101	001101	0101
D13.2	010	01101	101100	0101	101100	0101
D14.2	010	01110	011100	0101	011100	0101
D15.2	010	01111	010111	0101	101000	0101
D16.2	010	10000	011011	0101	100100	0101
D17.2	010	10001	100011	0101	100011	0101
D18.2	010	10010	010011	0101	010011	0101
D19.2	010	10011	110010	0101	110010	0101
D20.2	010	10100	001011	0101	001011	0101
D21.2	010	10101	101010	0101	101010	0101
D22.2	010	10110	011010	0101	011010	0101
D23.2	010	10111	111010	0101	000101	0101
D24.2	010	11000	110011	0101	001100	0101
D25.2	010	11001	100110	0101	100110	0101
D26.2	010	11010	010110	0101	010110	0101
D27.2	010	11011	110110	0101	001001	0101
D28.2	010	11100	001110	0101	001110	0101
D29.2	010	11101	101110	0101	010001	0101
D30.2	010	11110	011110	0101	100001	0101
D31.2	010	11111	101011	0101	010100	0101

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fghj	abcdei	fghj
D0.3	011	00000	100111	0011	011000	1100
D1.3	011	00001	011101	0011	100010	1100
D2.3	011	00010	101101	0011	010010	1100
D3.3	011	00011	110001	1100	110001	0011
D4.3	011	00100	110101	0011	001010	1100
D5.3	011	00101	101001	1100	101001	0011
D6.3	011	00110	011001	1100	011001	0011
D7.3	011	00111	111000	1100	000111	0011
D8.3	011	01000	111001	0011	000110	1100
D9.3	011	01001	100101	1100	100101	0011
D10.3	011	01010	010101	1100	010101	0011
D11.3	011	01011	110100	1100	110100	0011
D12.3	011	01100	001101	1100	001101	0011
D13.3	011	01101	101100	1100	101100	0011
D14.3	011	01110	011100	1100	011100	0011
D15.3	011	01111	010111	0011	101000	1100
D16.3	011	10000	011011	0011	100100	1100
D17.3	011	10001	100011	1100	100011	0011
D18.3	011	10010	010011	1100	010011	0011
D19.3	011	10011	110010	1100	110010	0011
D20.3	011	10100	001011	1100	001011	0011
D21.3	011	10101	101010	1100	101010	0011
D22.3	011	10110	011010	1100	011010	0011
D23.3	011	10111	111010	0011	000101	1100
D24.3	011	11000	110011	0011	001100	1100
D25.3	011	11001	100110	1100	100110	0011
D26.3	011	11010	010110	1100	010110	0011
D27.3	011	11011	110110	0011	001001	1100
D28.3	011	11100	001110	1100	001110	0011
D29.3	011	11101	101110	0011	010001	1100
D30.3	011	11110	011110	0011	100001	1100
D31.3	011	11111	101011	0011	010100	1100

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fghj	abcdei	fghj
D0.4	100	00000	100111	0010	011000	1101
D1.4	100	00001	011101	0010	100010	1101
D2.4	100	00010	101101	0010	010010	1101
D3.4	100	00011	110001	1101	110001	0010
D4.4	100	00100	110101	0010	001010	1101
D5.4	100	00101	101001	1101	101001	0010
D6.4	100	00110	011001	1101	011001	0010
D7.4	100	00111	111000	1101	000111	0010
D8.4	100	01000	111001	0010	000110	1101
D9.4	100	01001	100101	1101	100101	0010
D10.4	100	01010	010101	1101	010101	0010
D11.4	100	01011	110100	1101	110100	0010
D12.4	100	01100	001101	1101	001101	0010
D13.4	100	01101	101100	1101	101100	0010
D14.4	100	01110	011100	1101	011100	0010
D15.4	100	01111	010111	0010	101000	1101
D16.4	100	10000	011011	0010	100100	1101
D17.4	100	10001	100011	1101	100011	0010
D18.4	100	10010	010011	1101	010011	0010
D19.4	100	10011	110010	1101	110010	0010
D20.4	100	10100	001011	1101	001011	0010
D21.4	100	10101	101010	1101	101010	0010
D22.4	100	10110	011010	1101	011010	0010
D23.4	100	10111	111010	0010	000101	1101
D24.4	100	11000	110011	0010	001100	1101
D25.4	100	11001	100110	1101	100110	0010
D26.4	100	11010	010110	1101	010110	0010
D27.4	100	11011	110110	0010	001001	1101
D28.4	100	11100	001110	1101	001110	0010
D29.4	100	11101	101110	0010	010001	1101
D30.4	100	11110	011110	0010	100001	1101
D31.4	100	11111	101011	0010	010100	1101

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fg hj	abcdei	fg hj
D0.5	101	00000	100111	1010	011000	1010
D1.5	101	00001	011101	1010	100010	1010
D2.5	101	00010	101101	1010	010010	1010
D3.5	101	00011	110001	1010	110001	1010
D4.5	101	00100	110101	1010	001010	1010
D5.5	101	00101	101001	1010	101001	1010
D6.5	101	00110	011001	1010	011001	1010
D7.5	101	00111	111000	1010	000111	1010
D8.5	101	01000	111001	1010	000110	1010
D9.5	101	01001	100101	1010	100101	1010
D10.5	101	01010	010101	1010	010101	1010
D11.5	101	01011	110100	1010	110100	1010
D12.5	101	01100	001101	1010	001101	1010
D13.5	101	01101	101100	1010	101100	1010
D14.5	101	01110	011100	1010	011100	1010
D15.5	101	01111	010111	1010	101000	1010
D16.5	101	10000	011011	1010	100100	1010
D17.5	101	10001	100011	1010	100011	1010
D18.5	101	10010	010011	1010	010011	1010
D19.5	101	10011	110010	1010	110010	1010
D20.5	101	10100	001011	1010	001011	1010
D21.5	101	10101	101010	1010	101010	1010
D22.5	101	10110	011010	1010	011010	1010
D23.5	101	10111	111010	1010	000101	1010
D24.5	101	11000	110011	1010	001100	1010
D25.5	101	11001	100110	1010	100110	1010
D26.5	101	11010	010110	1010	010110	1010
D27.5	101	11011	110110	1010	001001	1010
D28.5	101	11100	001110	1010	001110	1010
D29.5	101	11101	101110	1010	010001	1010
D30.5	101	11110	011110	1010	100001	1010
D31.5	101	11111	101011	1010	010100	1010
D0.6	110	00000	100111	0110	011000	0110
D1.6	110	00001	011101	0110	100010	0110
D2.6	110	00010	101101	0110	010010	0110

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fghj	abcdei	fghj
D3.6	110	00011	110001	0110	110001	0110
D4.6	110	00100	110101	0110	001010	0110
D5.6	110	00101	101001	0110	101001	0110
D6.6	110	00110	011001	0110	011001	0110
D7.6	110	00111	111000	0110	000111	0110
D8.6	110	01000	111001	0110	000110	0110
D9.6	110	01001	100101	0110	100101	0110
D10.6	110	01010	010101	0110	010101	0110
D11.6	110	01011	110100	0110	110100	0110
D12.6	110	01100	001101	0110	001101	0110
D13.6	110	01101	101100	0110	101100	0110
D14.6	110	01110	011100	0110	011100	0110
D15.6	110	01111	010111	0110	101000	0110
D16.6	110	10000	011011	0110	100100	0110
D17.6	110	10001	100011	0110	100011	0110
D18.6	110	10010	010011	0110	010011	0110
D19.6	110	10011	110010	0110	110010	0110
D20.6	110	10100	001011	0110	001011	0110
D21.6	110	10101	101010	0110	101010	0110
D22.6	110	10110	011010	0110	011010	0110
D23.6	110	10111	111010	0110	000101	0110
D24.6	110	11000	110011	0110	001100	0110
D25.6	110	11001	100110	0110	100110	0110
D26.6	110	11010	010110	0110	010110	0110
D27.6	110	11011	110110	0110	001001	0110
D28.6	110	11100	001110	0110	001110	0110
D29.6	110	11101	101110	0110	010001	0110
D30.6	110	11110	011110	0110	100001	0110
D31.6	110	11111	101011	0110	010100	0110

Data Byte Name	Bits		Current RD-		Current RD+	
	HGF	EDCBA	abcdei	fghj	abcdei	fghj
D0.7	111	00000	100111	0001	011000	1110
D1.7	111	00001	011101	0001	100010	1110
D2.7	111	00010	101101	0001	010010	1110
D3.7	111	00011	110001	1110	110001	0001
D4.7	111	00100	110101	0001	001010	1110
D5.7	111	00101	101001	1110	101001	0001
D6.7	111	00110	011001	1110	011001	0001
D7.7	111	00111	111000	1110	000111	0001
D8.7	111	01000	111001	0001	000110	1110
D9.7	111	01001	100101	1110	100101	0001
D10.7	111	01010	010101	1110	010101	0001
D11.7	111	01011	110100	1110	110100	1000
D12.7	111	01100	001101	1110	001101	0001
D13.7	111	01101	101100	1110	101100	1000
D14.7	111	01110	011100	1110	011100	1000
D15.7	111	01111	010111	0001	101000	1110
D16.7	111	10000	011011	0001	100100	1110
D17.7	111	10001	100011	0111	100011	0001
D18.7	111	10010	010011	0111	010011	0001
D19.7	111	10011	110010	1110	110010	0001
D20.7	111	10100	001011	0111	001011	0001
D21.7	111	10101	101010	1110	101010	0001
D22.7	111	10110	011010	1110	011010	0001
D23.7	111	10111	111010	0001	000101	1110
D24.7	111	11000	110011	0001	001100	1110
D25.7	111	11001	100110	1110	100110	0001
D26.7	111	11010	010110	1110	010110	0001
D27.7	111	11011	110110	0001	001001	1110
D28.7	111	11100	001110	1110	001110	0001
D29.7	111	11101	101110	0001	010001	1110
D30.7	111	11110	011110	0001	100001	1110
D31.7	111	11111	101011	0001	010100	1110

## A2 Valid special characters

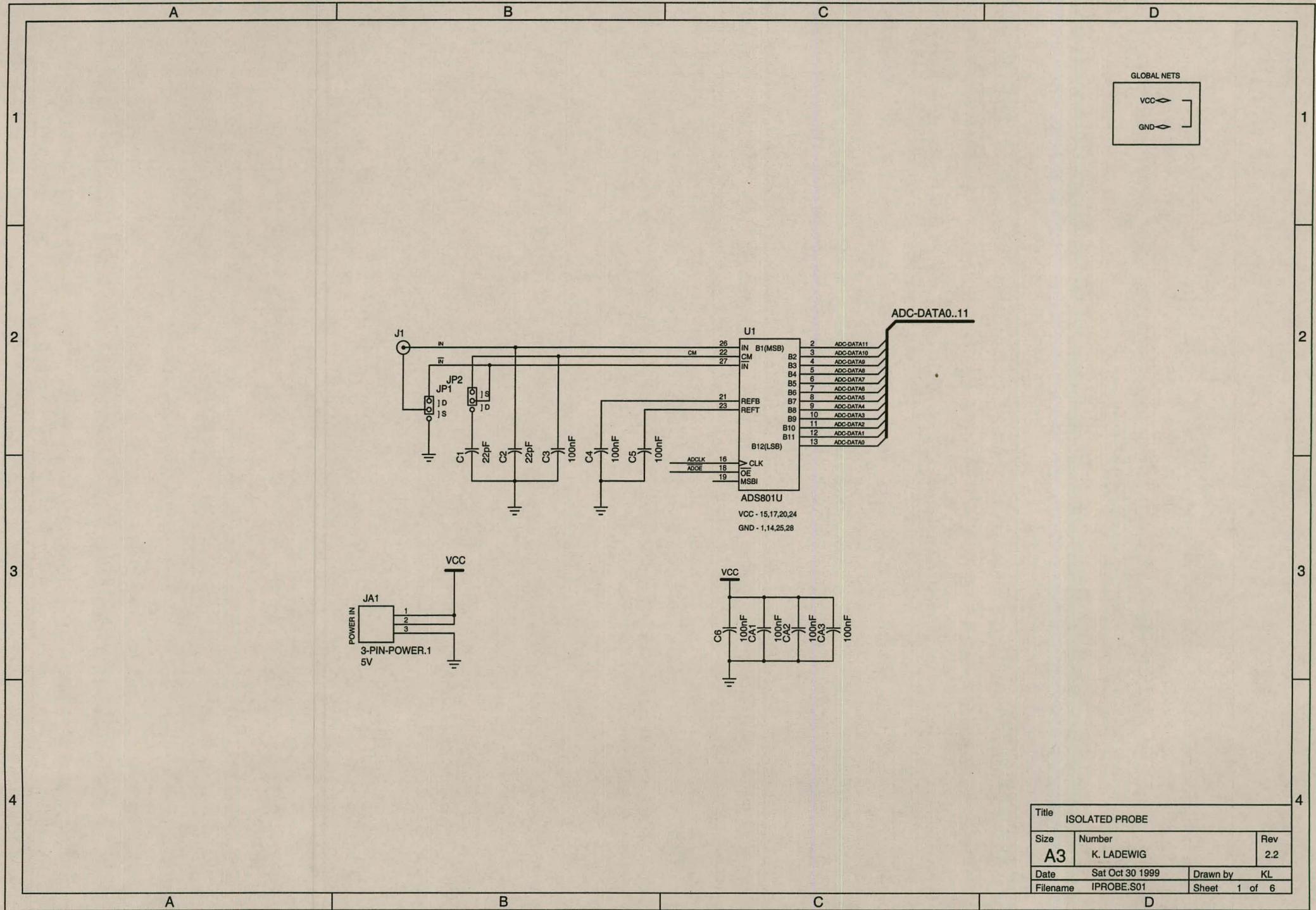
S.C. Byte Name	S.C. Code Name	Bits		Current RD-		Current RD+		
		HGF	EDCBA	abcdei	fg hj	abcdei	fg hj	
K28.0	C0.0 (C00)	000	00000	001111	0100	110000	1011	
K28.1	C1.0 (C01)	000	00001	001111	1001	110000	0110	
K28.2	C2.0 (C02)	000	00010	001111	0101	110000	1010	
K28.3	C3.0 (C03)	000	00011	001111	0011	110000	1100	
K28.4	C4.0 (C04)	000	00100	001111	0010	110000	1101	
K28.5	C5.0 (C05)	000	00101	001111	1010	110000	0101	
K28.6	C6.0 (C06)	000	00110	001111	0110	110000	1001	
K28.7	C7.0 (C07)	000	00111	001111	1000	110000	0111	
K23.7	C8.0 (C08)	000	01000	110101	1000	000101	0111	
K27.7	C9.0 (C09)	000	01001	110110	1000	001001	0111	
K29.7	C10.0 (C0A)	000	01010	101110	1000	010001	0111	
K30.7	C11.0 (C0B)	000	01011	011110	1000	100001	0111	
Idle	C0.1 (C20)	001	00000	-K28.5+,D21.4,D21.5,D21.5,repeat <sup>[25]</sup>				
R_RDY	C1.1 (C21)	001	00001	-K28.5+,D21.4,D10.2,D10.2,repeat <sup>[26]</sup>				
EOFxx	C2.1 (C22)	001	00010	-K28.5,Dn.xxx0 <sup>[27]</sup> +K28.5,Dn.xxx1 <sup>[27]</sup>				
	Follows K28.1 for ESCON Connect-SOF (Rx indication only)							
C-SOF	C7.1 (C27)	001	00111	001111	1000	110000	0111	
	Follows K28.5 for ESCON Passive-SOF (Rx indication only)							
P-SOF	C7.2 (C47)	010	00111	001111	1000	110000	0111	
	Code Rule Violation and SVS Tx Pattern							
Exception	C0.7 (CE0)	111	00000	100111	1000 <sup>[28]</sup>	011000	0111 <sup>[28]</sup>	
-K28.5	C1.7 (CE1)	111	00001	001111	1010 <sup>[29]</sup>	001111	1010 <sup>[29]</sup>	
+K28.5	C2.7 (CE2)	111	00010	110000	0101 <sup>[30]</sup>	110000	0101 <sup>[30]</sup>	
	Running Disparity Violation Pattern							
Exception	C4.7 (CE4)	111	00100	110111	0101 <sup>[31]</sup>	001000	1010 <sup>[31]</sup>	

## Notes:

23. All codes not shown are reserved.
24. Notation for Special Character Byte Name is consistent with Fibre Channel and ESCON naming conventions. Special Character Code Name is intended to describe binary information present on I/O pins. Common usage for the name can either be in the form used for describing Data patterns (i.e., C0.0 through C31.7), or in hex notation (i.e., Cnn where nn=the specified value between 00 and FF).
25. C0.1 = Transmit Negative K28.5 (-K28.5+) disregarding Current RD when input is held for only one byte time. If held longer, transmitter begins sending the repeating transmit sequence -K28.5+, D21.4, D21.5, D21.5, (repeat all four bytes)... defined in X3.230 as the primitive signal "Idle word." This Special Character input must be held for four (4) byte times or multiples of four bytes or it will be truncated by the new data. The receiver will never output this Special Character, since K28.5 is decoded as C5.0, C1.7, or C2.7, and the subsequent bytes are decoded as data.
26. C1.1 = Transmit Negative K28.5 (-K28.5+) disregarding Current RD when input is held for only one byte time. If held longer, transmitter begins sending the repeating transmit sequence -K28.5+, D21.4, D10.2, D10.2,(repeat all four bytes)... defined in X3.230 as the primitive signal "Receiver\_Ready (R\_RDY)." This Special Character input must be held for four (4) byte times or multiples of four bytes or it will be truncated by the new data.
27. C2.1 = Transmit either -K28.5+ or +K28.5- as determined by Current RD and modify the Transmission Character that follows, by setting its least significant bit to 1 or 0. If Current RD at the start of the following character is plus (+) the LSB is set to 0, and if Current RD is minus (-) the LSB becomes 1. This modification allows construction of X3.230 "EOF" frame delimiters wherein the second data byte is determined by the Current RD. For example, to send "EOFdt" the controller could issue the sequence C2.1-D21.4- D21.4-D21.4, and the HOTLink Transmitter will send either K28.5-D21.4-D21.4-D21.4 or K28.5-D21.5- D21.4-D21.4 based on Current RD. Likewise to send "EOFdti" the controller could issue the sequence C2.1-D10.4-D21.4-D21.4, and the HOTLink Transmitter will send either K28.5-D10.4-D21.4- D21.4 or K28.5-D10.5-D21.4- D21.4 based on Current RD. The receiver will never output this Special Character, since K28.5 is decoded as C5.0, C1.7, or C2.7, and the subsequent bytes are decoded as data.
28. C0.7 = Transmit a deliberate code rule violation. The code chosen for this function follows the normal Running Disparity rules. Transmission of this Special Character has the same effect as asserting SVS = HIGH. The receiver will only output this Special Character if the Transmission Character being decoded is not found in the tables.

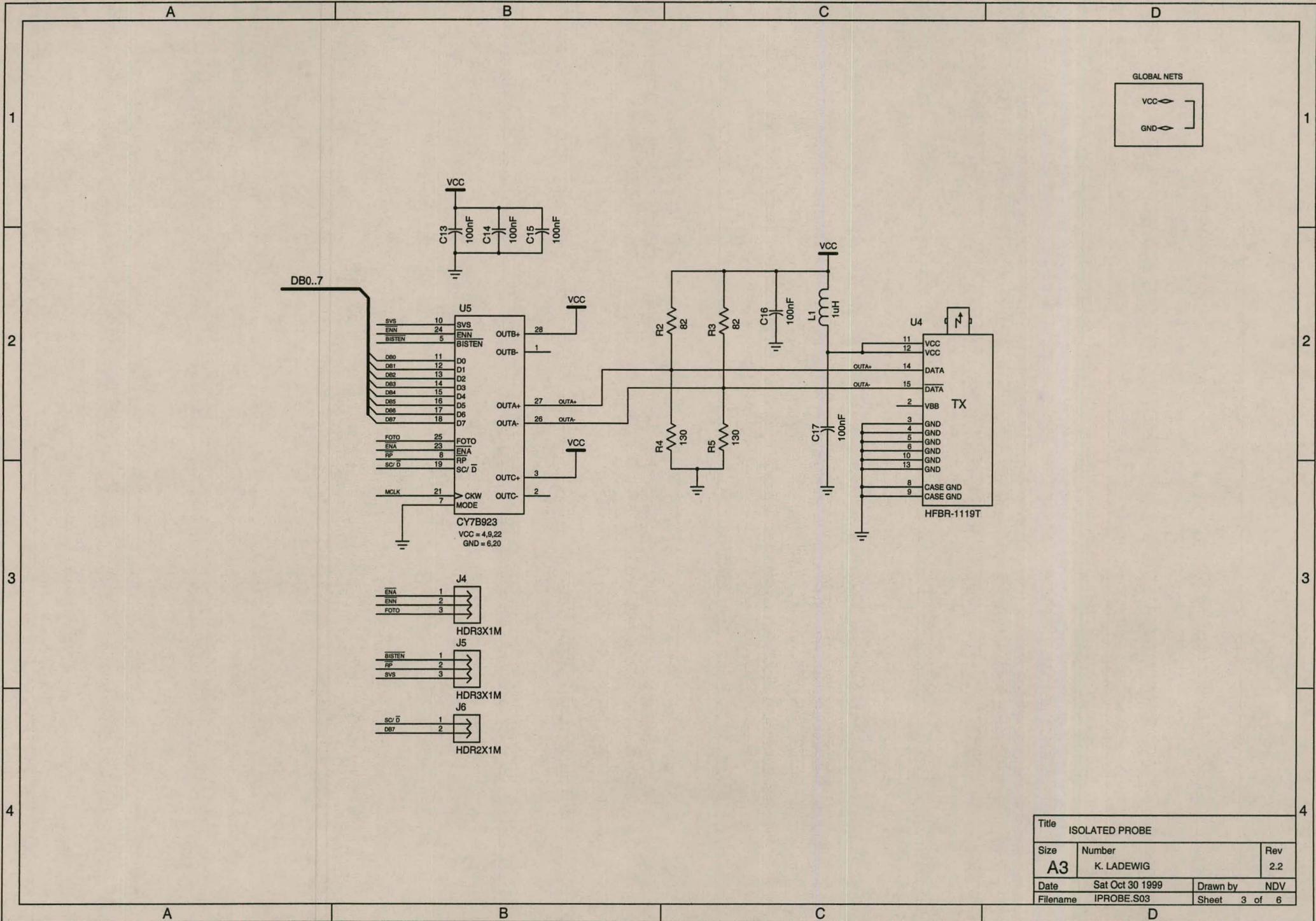
## **Appendix B**

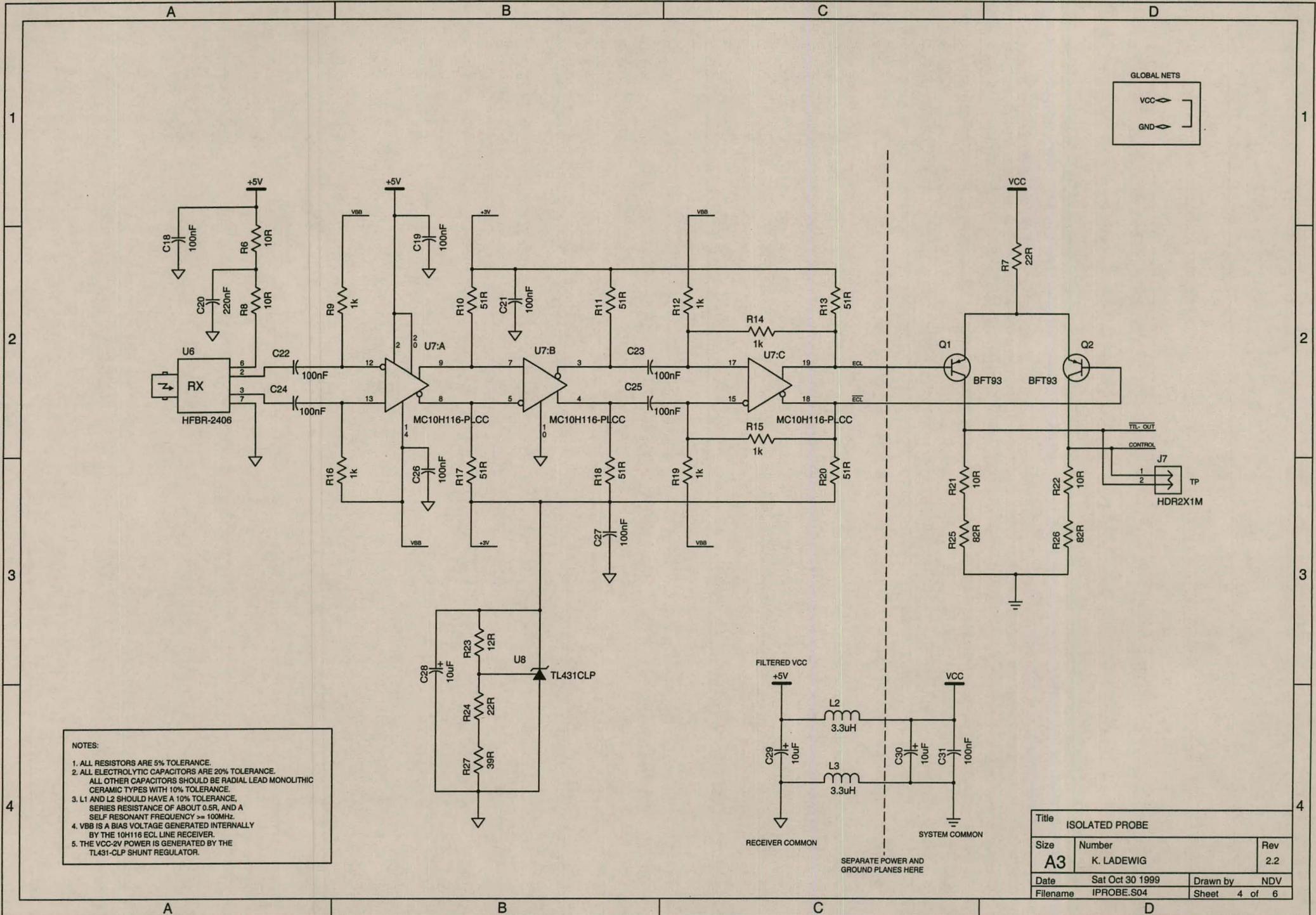
### **Schematic diagrams for isolated probe**



Title ISOLATED PROBE		
Size <b>A3</b>	Number K. LADEWIG	Rev 2.2
Date Sat Oct 30 1999	Drawn by KL	
Filename IPROBE.S01	Sheet 1 of 6	



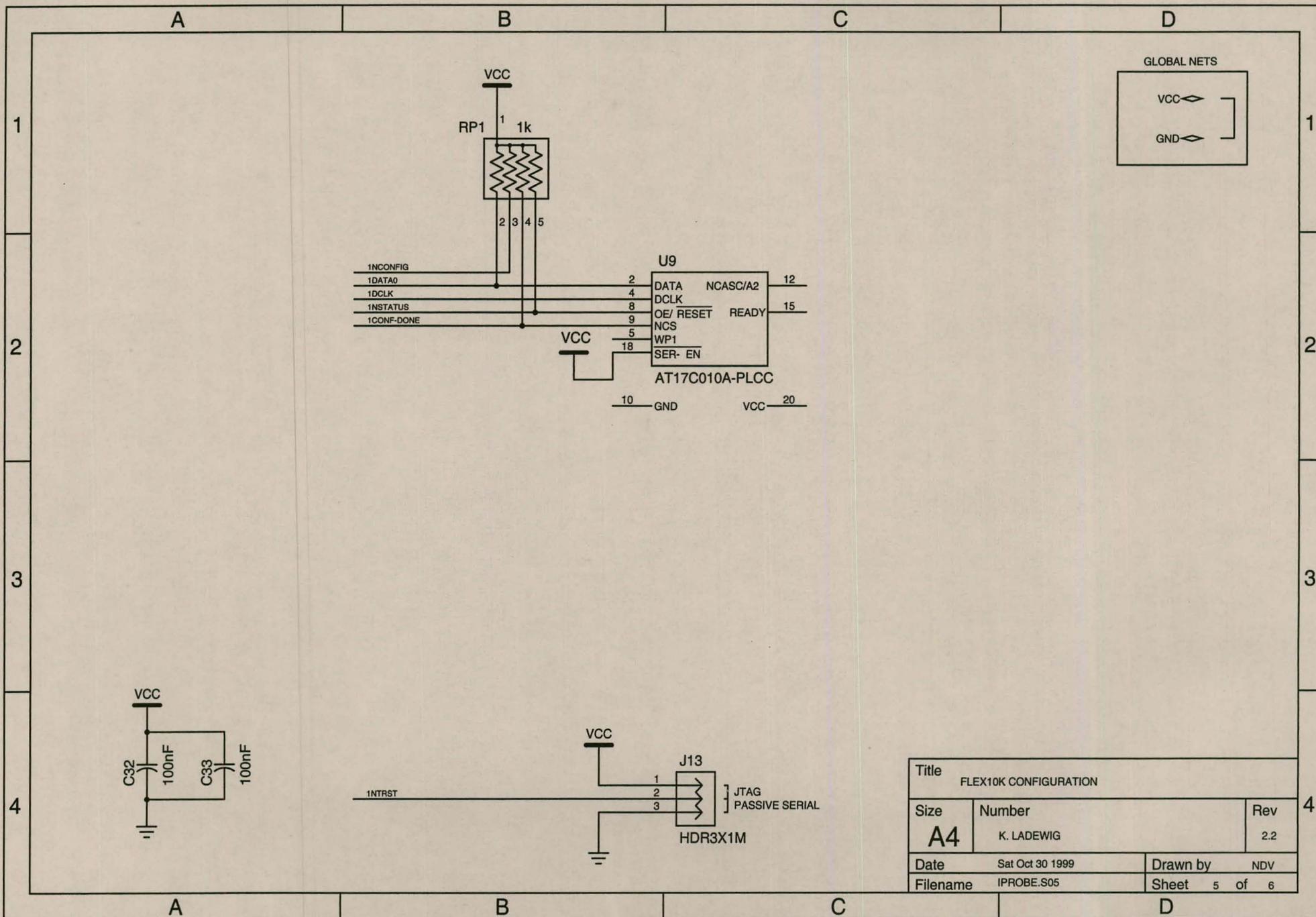




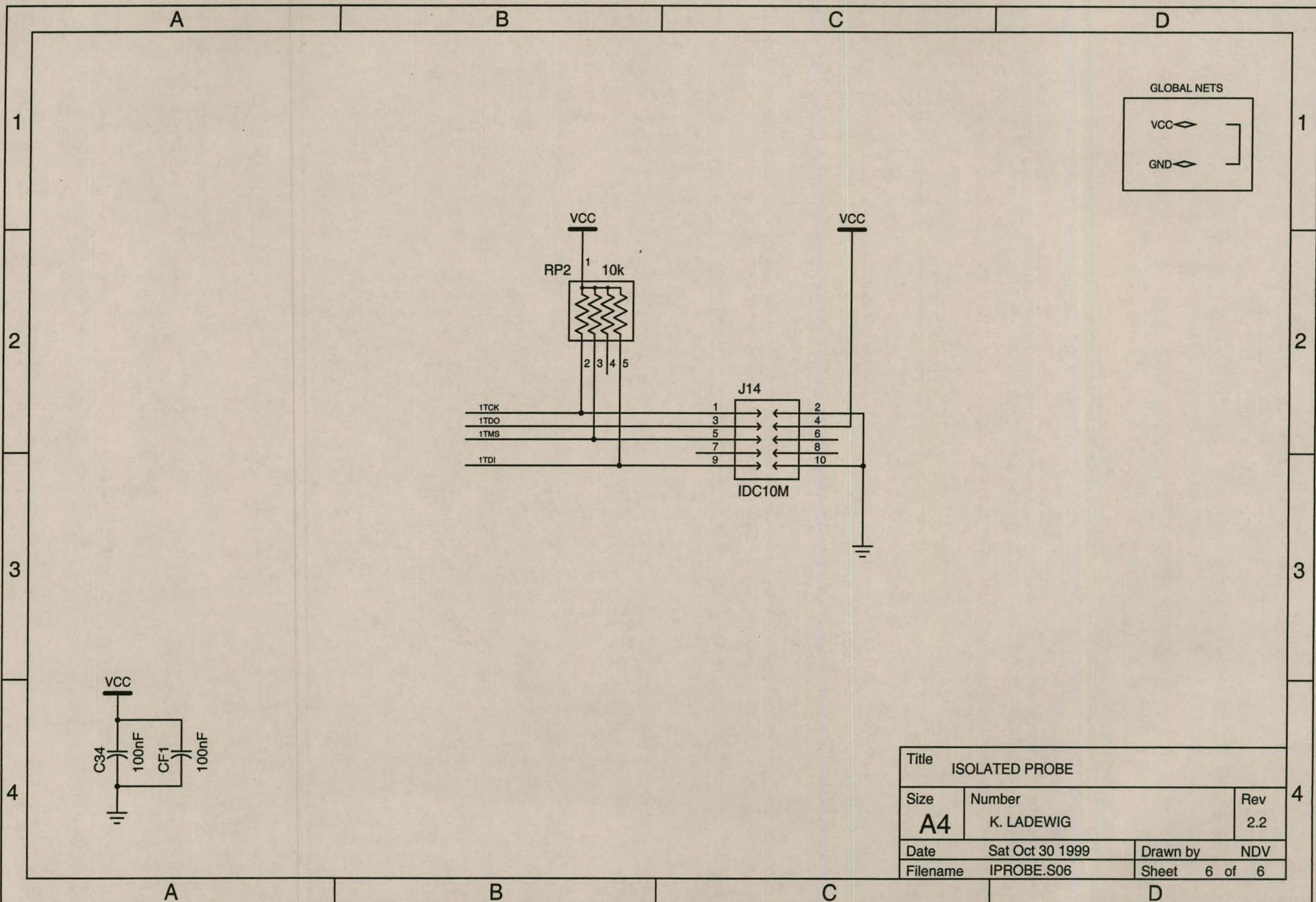
NOTES:

1. ALL RESISTORS ARE 5% TOLERANCE.
2. ALL ELECTROLYTIC CAPACITORS ARE 20% TOLERANCE. ALL OTHER CAPACITORS SHOULD BE RADIAL LEAD MONOLITHIC CERAMIC TYPES WITH 10% TOLERANCE.
3. L1 AND L2 SHOULD HAVE A 10% TOLERANCE, SERIES RESISTANCE OF ABOUT 0.5R, AND A SELF RESONANT FREQUENCY  $\gg 100$ MHz.
4. VBB IS A BIAS VOLTAGE GENERATED INTERNALLY BY THE 10H116 ECL LINE RECEIVER.
5. THE VCC-2V POWER IS GENERATED BY THE TL431-CLP SHUNT REGULATOR.

Title ISOLATED PROBE		
Size A3	Number K. LADEWIG	Rev 2.2
Date Sat Oct 30 1999	Drawn by NDV	
Filename IPROBE.S04	Sheet 4 of 6	

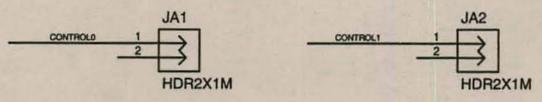
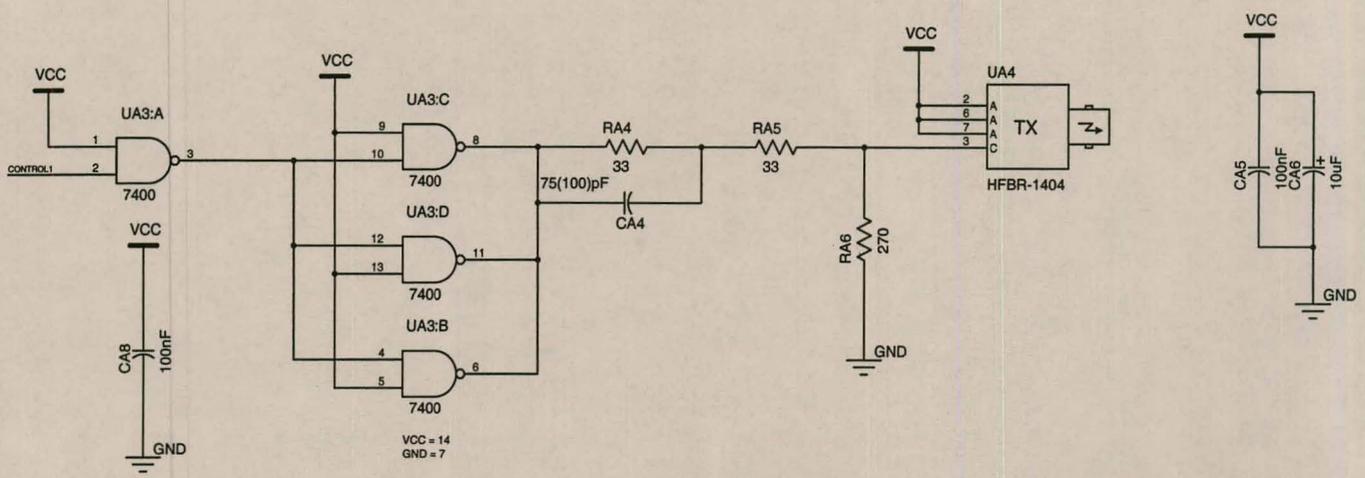
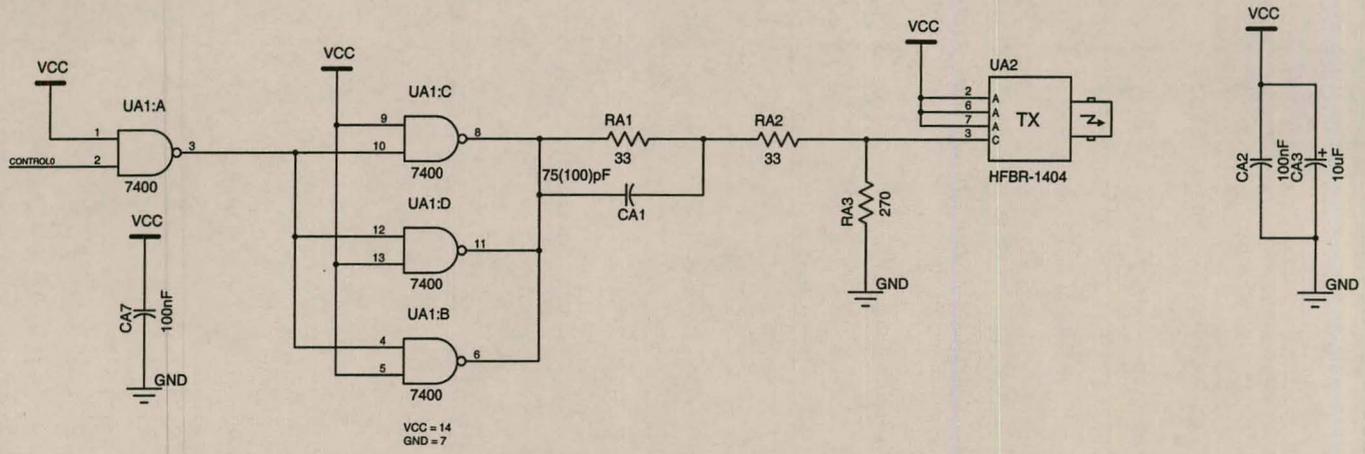


Title		
FLEX10K CONFIGURATION		
Size	Number	Rev
A4	K. LADEWIG	2.2
Date	Sat Oct 30 1999	Drawn by NDV
Filename	Iprobe.S05	Sheet 5 of 6



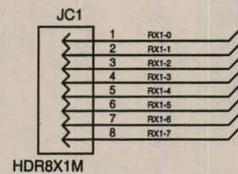
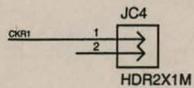
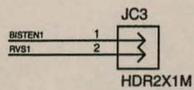
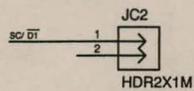
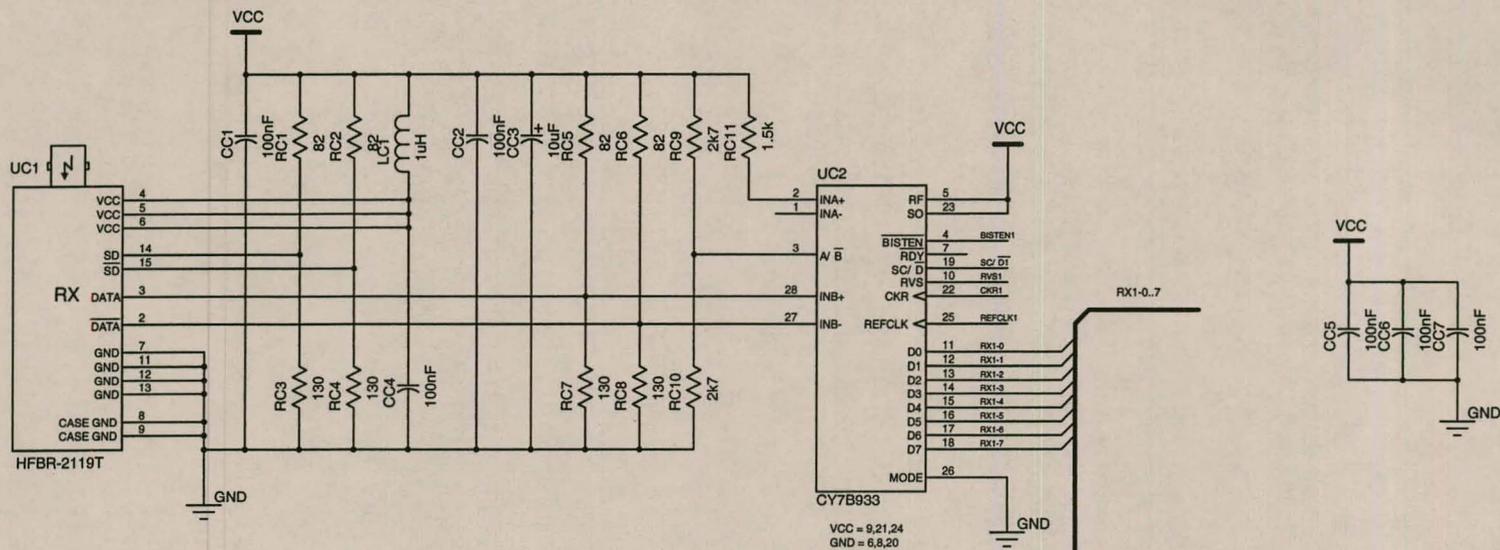
## **Appendix C**

### **Schematic diagrams for remote acquisition unit (RAU)**

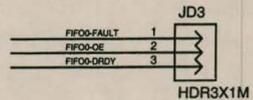
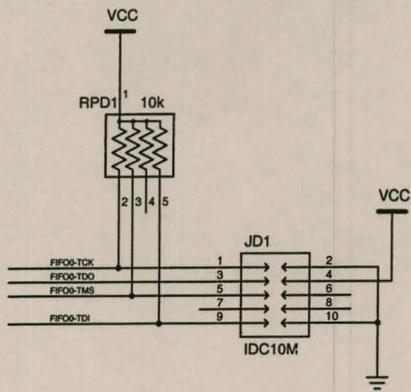


Title RAU		
Size A3	Number 155MBd TRANSMITTERS	Rev 1.0
Date Sat Oct 30 1999	Drawn by KL	
Filename RAIBORD.S01	Sheet 1 of 6	



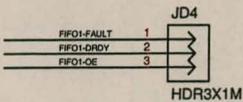
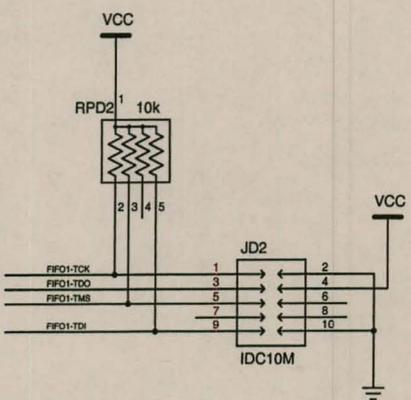
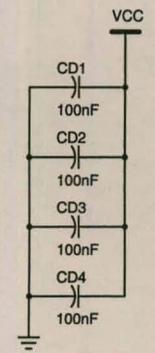
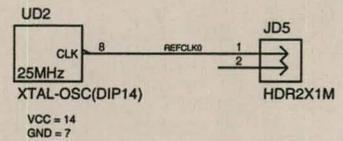


Title RAU		
Size A3	Number SECOND 266MBd TRANSMITTER	Rev 1.0
Date Sat Oct 30 1999	Drawn by KL	
Filename RAIBORD.S03	Sheet 3 of 6	



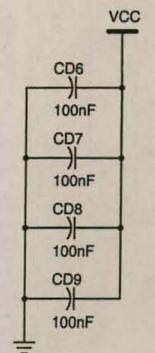
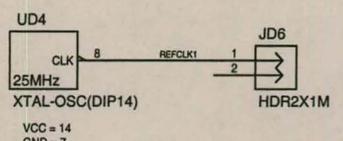
FIF00-TDI	7	TDI	CLR	1	RESET-OUT
FIF00-TMS	13	TMS	OE2	2	
FIF00-TCK	32	TCK	43	CKR0	
FIF00-TDO	38	TDO	44	FIF00-OE	
RAI-BUS0	4	I/O	41	RX0-1	
RAI-BUS2	5	I/O	40	RX0-0	
RAI-BUS1	6	I/O	39	RX0-2	
RAI-BUS3	8	I/O	37	RX0-3	
RAI-BUS4	9	I/O	36	RX0-4	
RAI-BUS6	11	I/O	34	RX0-5	
RAI-BUS5	12	I/O	33	RX0-6	
RAI-BUS7	14	I/O	31	RX0-7	
RAI-BUS8	16	I/O	29		
RAI-BUS10	17	I/O	28	SC/DT	
RAI-BUS12	18	I/O	27	FIF00-FAULT	
RAI-BUS9	19	I/O	26	FIF00-DRDY	
RAI-BUS13	20	I/O	25	RAI-BUS14	
RAI-BUS11	21	I/O	24	RAI-BUS15	

EPM7064-44  
-10  
VCC = 3,15,23,35  
GND = 10,22,30,42

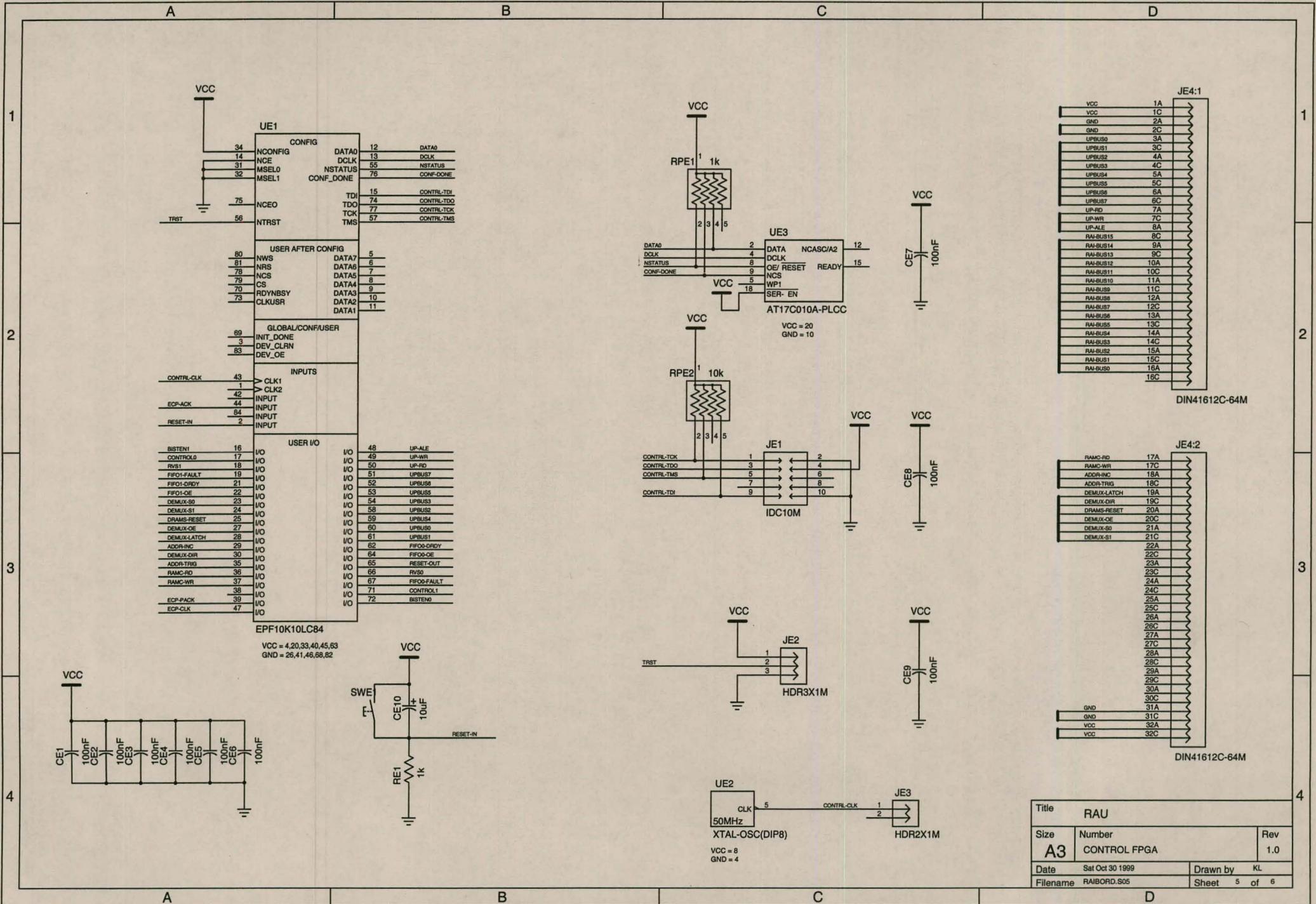


FIF01-TDI	7	TDI	CLR	1	RESET-OUT
FIF01-TMS	13	TMS	OE2	2	
FIF01-TCK	32	TCK	43	CKR1	
FIF01-TDO	38	TDO	44	FIF01-OE	
RAI-BUS0	4	I/O	41	RX1-1	
RAI-BUS2	5	I/O	40	RX1-0	
RAI-BUS1	6	I/O	39	RX1-2	
RAI-BUS3	8	I/O	37	RX1-3	
RAI-BUS4	9	I/O	36	RX1-4	
RAI-BUS5	11	I/O	34	RX1-5	
RAI-BUS6	12	I/O	33	RX1-6	
RAI-BUS7	14	I/O	31	RX1-7	
RAI-BUS8	16	I/O	29		
RAI-BUS10	17	I/O	28	SC/DT	
RAI-BUS12	18	I/O	27	FIF01-FAULT	
RAI-BUS9	19	I/O	26	FIF01-DRDY	
RAI-BUS13	20	I/O	25	RAI-BUS14	
RAI-BUS11	21	I/O	24	RAI-BUS15	

EPM7064-44  
-10  
VCC = 3,15,23,35  
GND = 10,22,30,42



Title RAU		
Size A3	Number FIF0'S IMPLEMENTED IN EPLD'S	Rev 1.0
Date Sat Oct 30 1999	Drawn by KL	
Filename RAIBORD.S04	Sheet 4	of 6



Title RAU		
Size A3	Number CONTROL FPGA	Rev 1.0
Date Set Oct 30 1999	Drawn by KL	
Filename RAIBORD.S05	Sheet 5 of 6	

UG1

MUX-TDI	7	TDI	1	CLR	2	ECP-CLK
MUX-TMS	13	TMS	OE2	43	10M-MUXCLK	
MUX-TCK	32	TCK	OE1	44	ECP-PACK	
MUX-TDO	38	TDO				
ECP-ACK	4	I/O	41	ECP6		
PERPHCLK	5	I/O	40	ECP7		
PERPHACK	6	I/O	39	ECP5		
RAI-BUS0	8	I/O	37	ECP3		
RAI-BUS1	9	I/O	36	ECP4		
RAI-BUS2	11	I/O	34	ECP1		
RAI-BUS3	12	I/O	33	ECP2		
RAI-BUS4	14	I/O	31	ECP0		
RAI-BUS6	16	I/O	29	HOSTACK		
RAI-BUS7	17	I/O	28	RAI-BUS15		
RAI-BUS7	18	I/O	27	RAI-BUS13		
RAI-BUS8	19	I/O	26	RAI-BUS14		
RAI-BUS9	20	I/O	25	RAI-BUS11		
RAI-BUS10	21	I/O	24	RAI-BUS12		

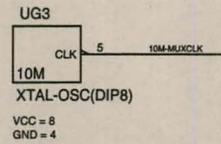
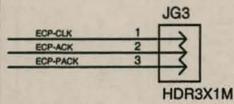
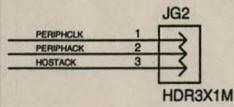
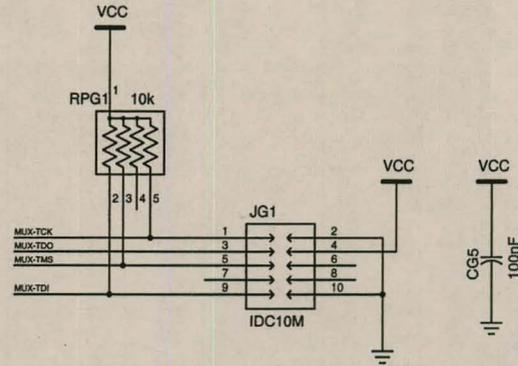
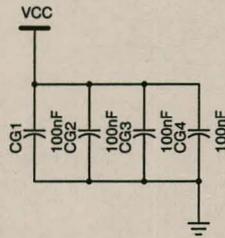
EPM7064-44  
-10  
VCC = 3,15,23,35  
GND = 10,22,30,42

JG4

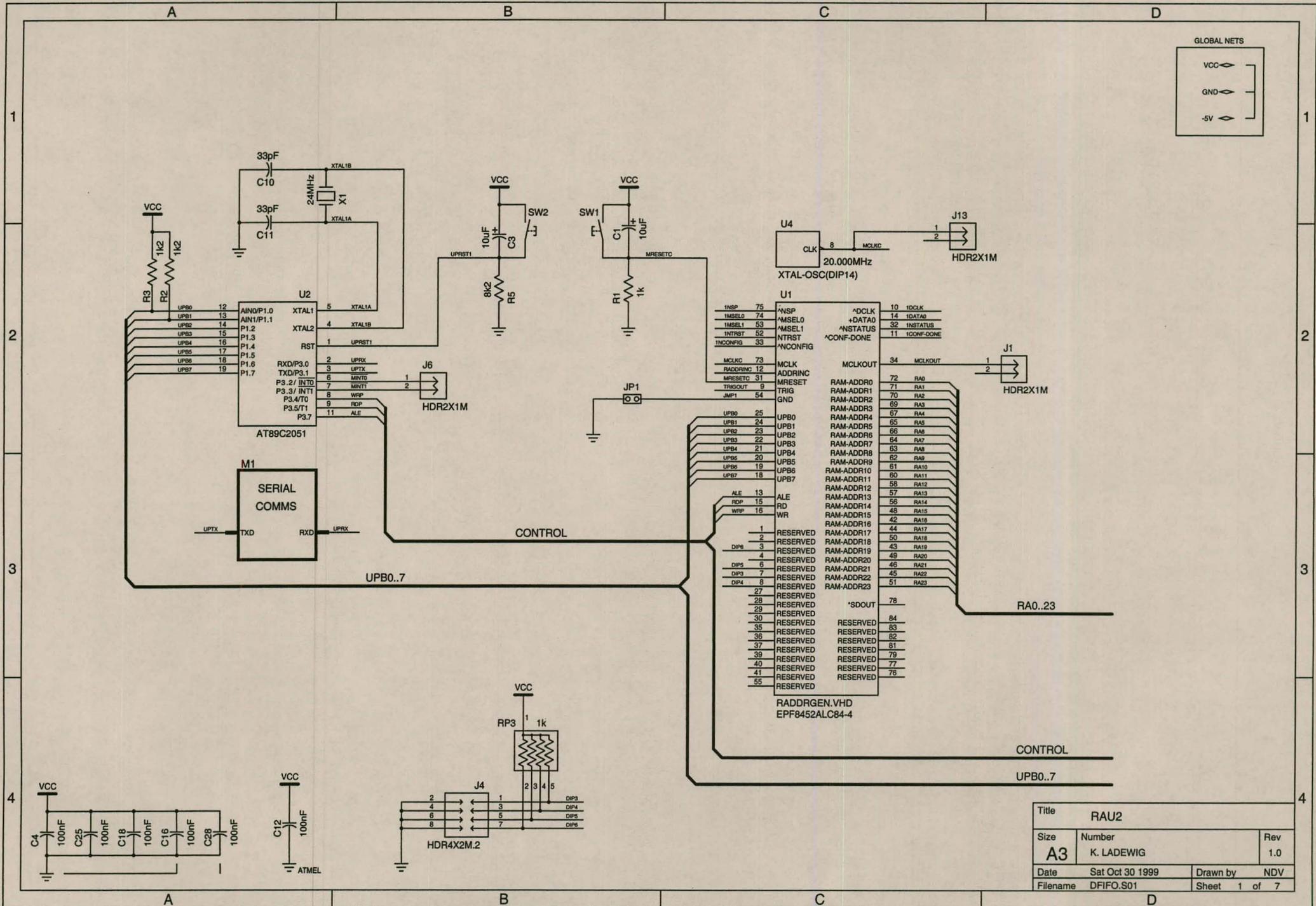
HOSTACK	1
ECP0	2
ECP1	3
ECP2	4
ECP3	5
ECP4	6
ECP5	7
ECP6	8
ECP7	9
PERPHCLK	10
PERPHACK	11

ECP-INTERFACE  
GND = 12,13,14

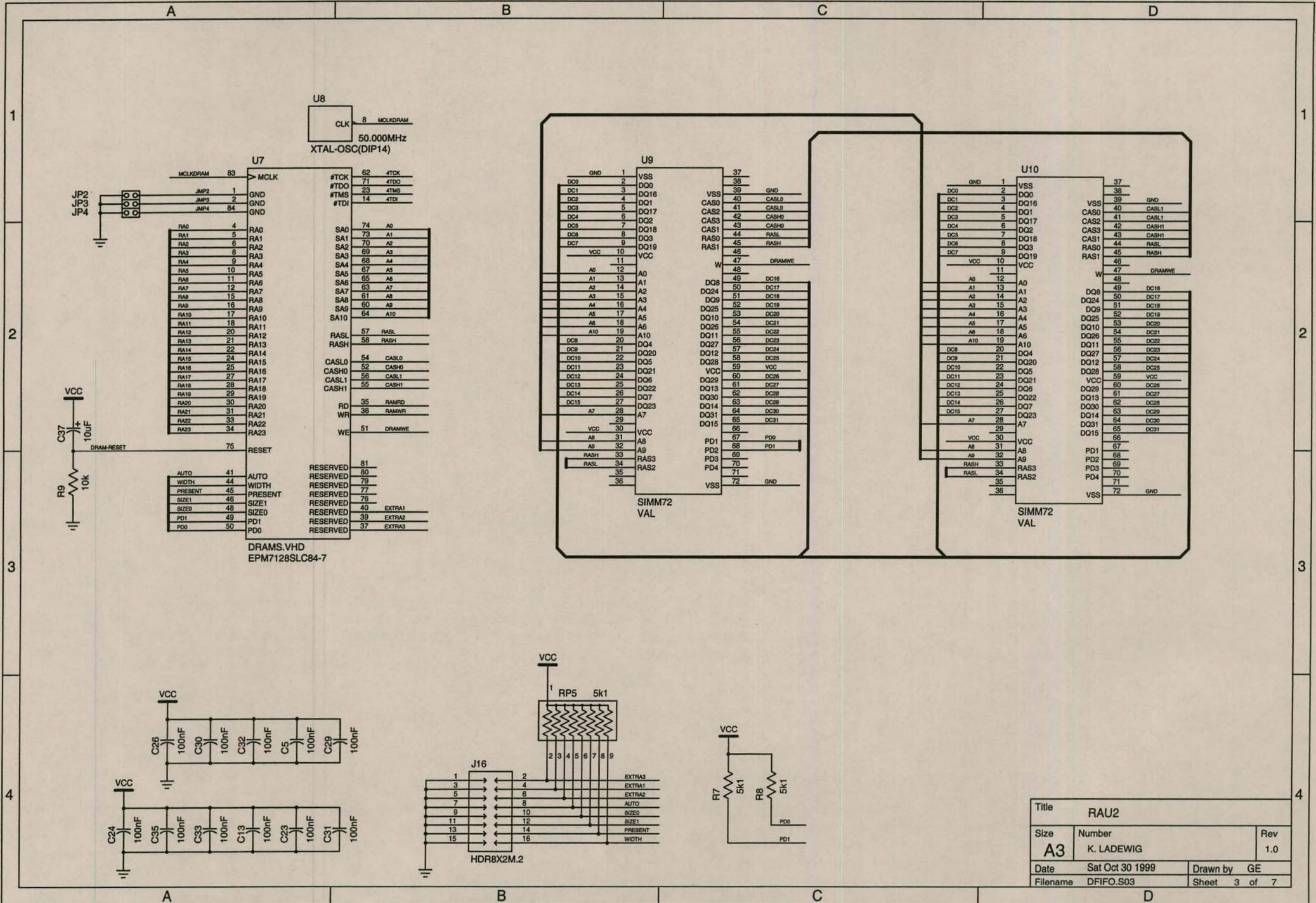
Note: In parallel port cable to PC nReverse Request (pin 16) must be connected to nAckReverse (pin 12).



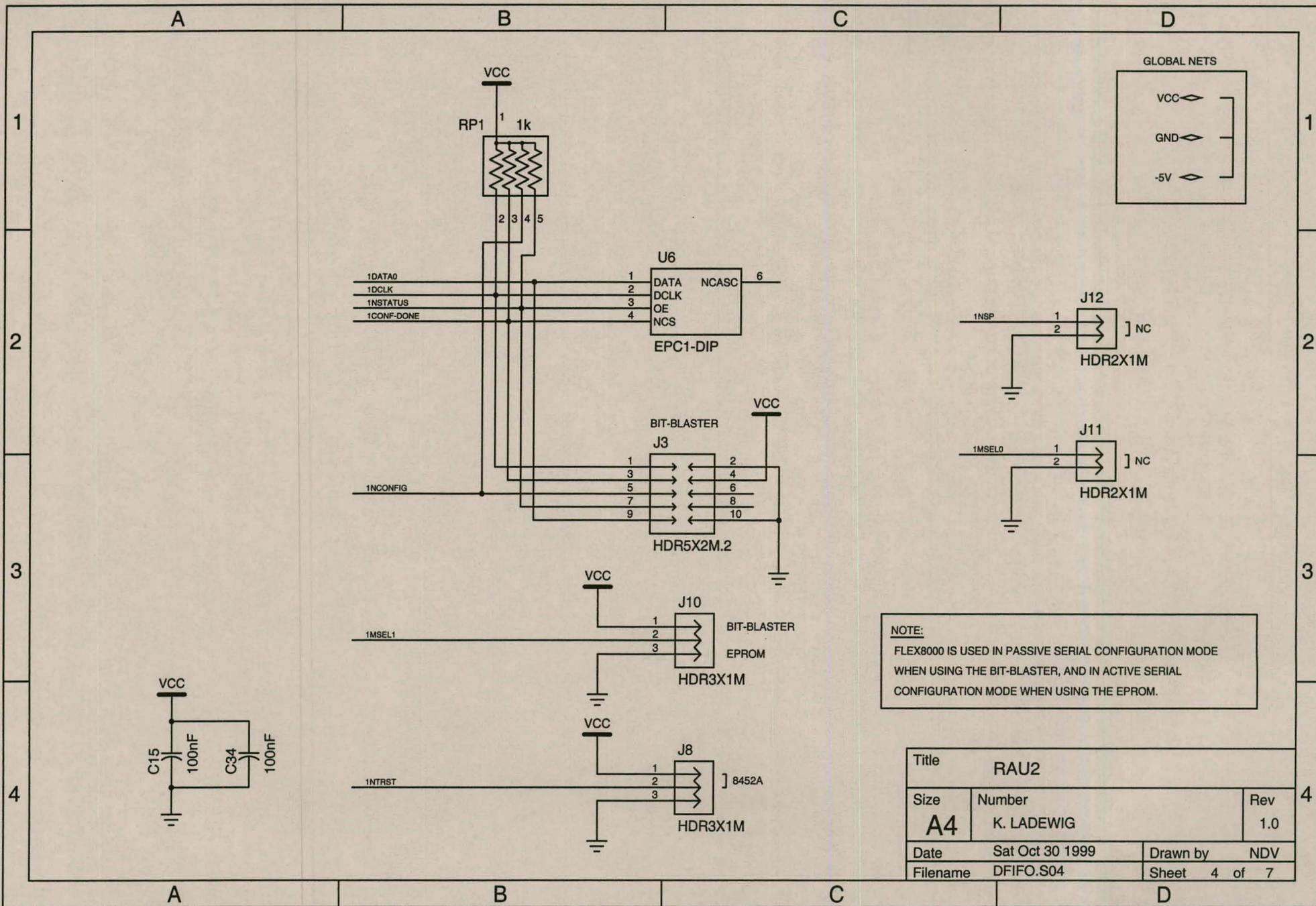
Title RAU		
Size A3	Number POSSIBLE ECP INTERFACE	Rev 1.0
Date Sat Oct 30 1999	Drawn by KL	
Filename RAIBORD.S06	Sheet 6 of 6	





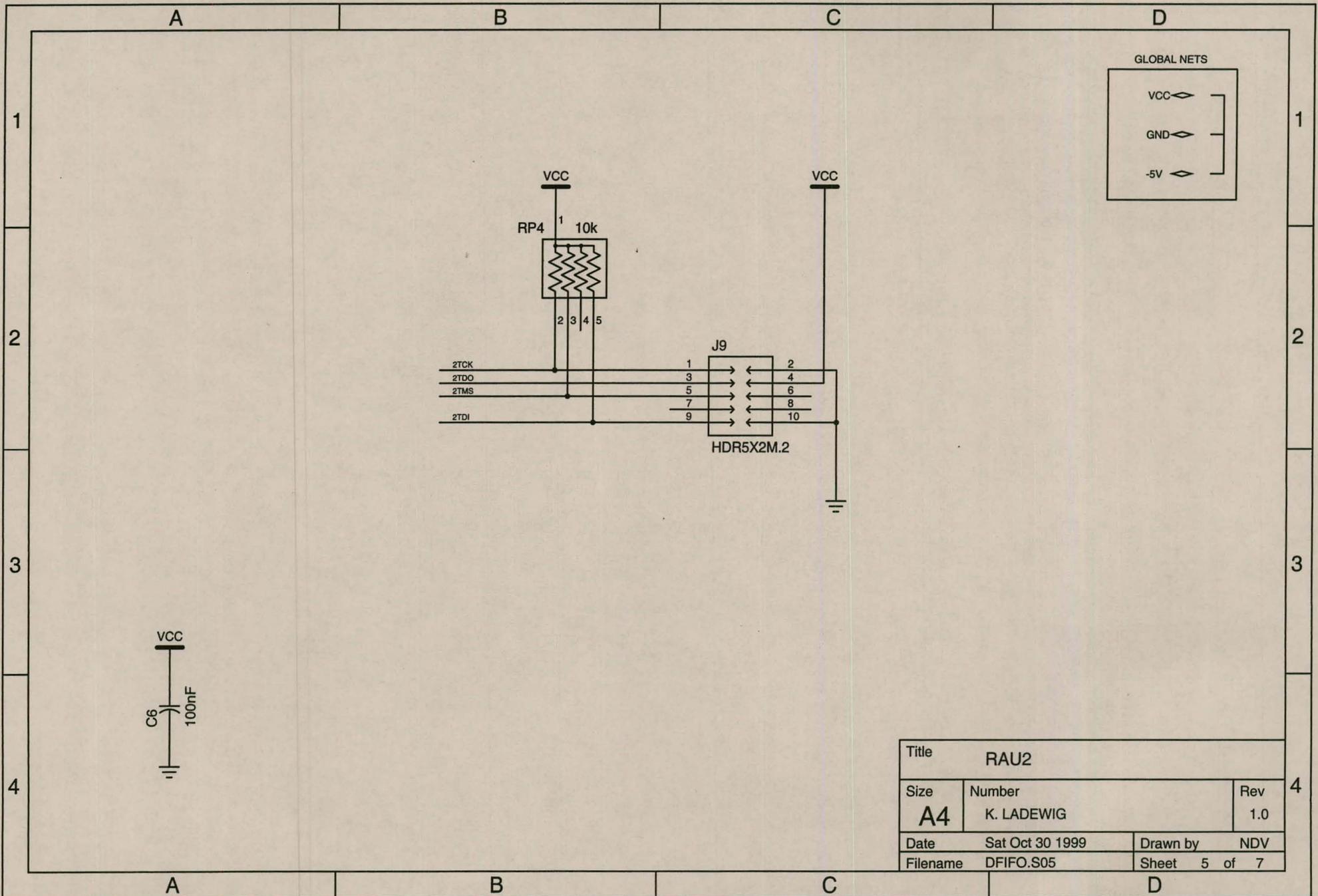


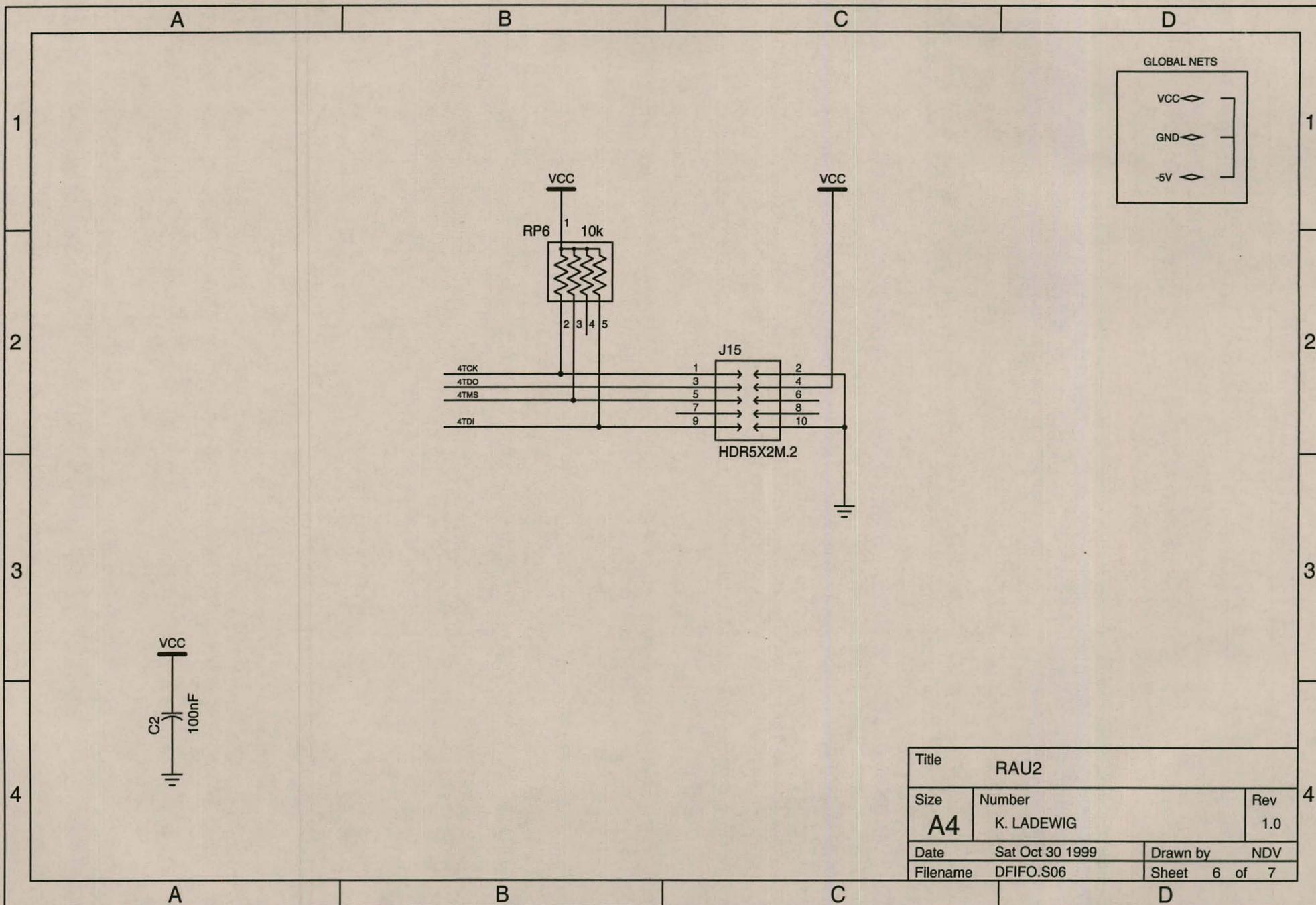
Title RAU2		
Size A3	Number K. LADEWIG	Rev 1.0
Date Sat Oct 30 1999	Drawn by GE	
Filename DFIFO.S03	Sheet 3 of 7	



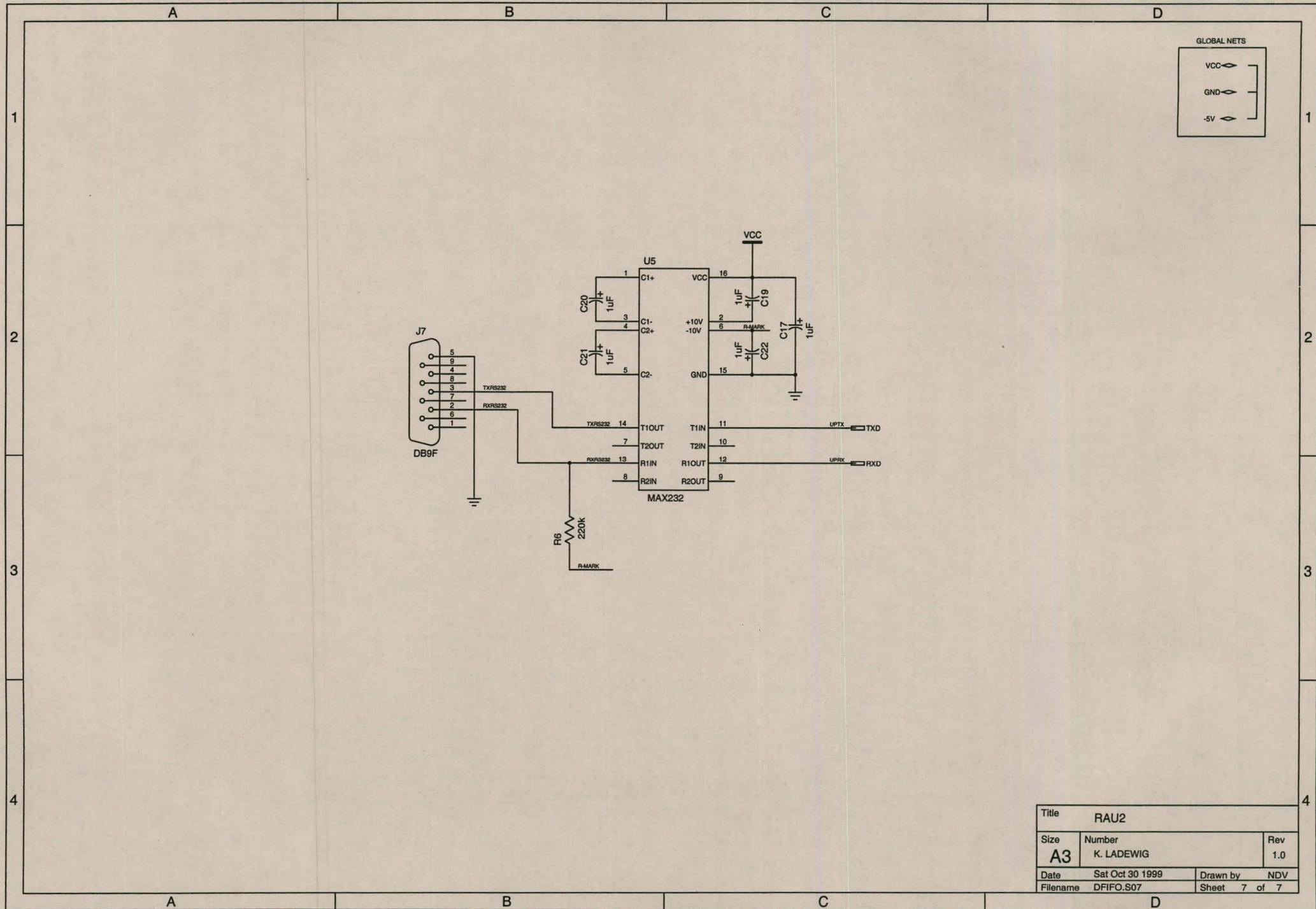
**NOTE:**  
 FLEX8000 IS USED IN PASSIVE SERIAL CONFIGURATION MODE  
 WHEN USING THE BIT-BLASTER, AND IN ACTIVE SERIAL  
 CONFIGURATION MODE WHEN USING THE EPROM.

Title			RAU2		
Size	Number			Rev	
<b>A4</b>	K. LADEWIG			1.0	
Date	Sat Oct 30 1999	Drawn by	NDV		
Filename	DFIFO.S04	Sheet	4 of 7		





Title			RAU2		
Size	Number			Rev	4
A4	K. LADEWIG			1.0	
Date	Sat Oct 30 1999	Drawn by	NDV		
Filename	DFIFO.S06	Sheet	6 of 7		



GLOBAL NETS

VCC	⎓
GND	⏏
-5V	⎓

Title		RAU2	
Size	Number	Rev	
A3	K. LADEWIG	1.0	
Date	Sat Oct 30 1999	Drawn by	NDV
Filename	DFIFO.S07	Sheet	7 of 7

## Appendix D

### VHDL-code

#### D1 Isolated probe control logic VHDL-code

```

library IEEE;
use IEEE.std_logic_1164.all;

entity TXFPGA is
port( -- Master clock signal
      Probe_mclk      : in std_logic;                -- 25 MHz clock signal from crystal oscillator

      -- Interface lines to ADC
      ADC_Data        : in std_logic_vector(11 downto 0); -- 12-bit digital output data from ADS801
      ADC_OE          : out std_logic;                -- ADC output-enable line
      ADC_Clock       : out std_logic;                -- ADC conversion clock

      -- Interface lines to CY7B923
      PS_Data         : out std_logic_vector(7 downto 0); -- 8-bit digital input data to CY7B923
      SC_D            : out std_logic;                -- special character/data line
      ENA             : out std_logic;                -- enable transmission line
      FOTO           : out std_logic;                -- fibre-optic turn off line
      SVS            : out std_logic;                -- send violation symbol
      BISTEN         : out std_logic;                -- built-in self test enable line
      ENN            : out std_logic;                -- enable transmission line

      -- RAU_clock line from RAI Unit
      RAU_clock       : in std_logic;                -- control from RAU (fibre optic control line)
end TXFPGA;

architecture A of TXFPGA is
  signal counter0      : integer range 1300 downto 0;
  signal change_to_acq : std_logic;
  signal OPERAT        : std_logic_vector(2 downto 0);
  signal counter2      : integer range 4 downto 0;
  signal Flag          : std_logic;
begin
  -- *****
  --                               SETTING OPERATING MODE
  -- *****

  -- Process that controls counter0. Counter0 is used to determine how long RAU_clock is driven low. If
  -- RAU_clock is low for longer than 50 us, the isolated probe switches to idle mode. Otherwise the
  -- RAU_clock input is routed directly to the ADC as ADC_Clock.
  process(Probe_mclk)
  begin
    if Probe_mclk'event and Probe_mclk = '1' then -- Execute process on each rising edge of Probe_mclk
      if RAU_clock = '0' then
        if counter0 < 1255 then
          counter0 <= counter0 + 1; -- Increment counter0 if RAU_clock LO
        end if;
      else
        counter0 <= 0; -- Set counter0 to zero if RAU_clock HI
      end if;
    end if;
  end process;

  -- Process that controls change_to_acq. As soon as RAU_clock switches HI, change_to_acq is pulsed HI

```

-- for one master clock period. This causes the isolated probe to change to acquisition mode.

```

process(Probe_mclk)
begin
    if Probe_mclk'event and Probe_mclk = '1' then          -- Execute process on each rising edge of Probe_mclk
        if OPERAT(1) = '1' then                            -- Idle mode
            if RAU_clock = '1' then
                change_to_acq <= '1';                    -- Change to acquisition mode when RAU_clock HI
            else
                change_to_acq <= '0';
            end if;
        else
            change_to_acq <= '0';
        end if;
    end if;
end process;

```

-- Process that controls OPERAT. OPERAT determines the operating mode of the isolated probe. It is derived..  
-- from counter0 and change\_to\_acq.

```

process(Probe_mclk)
begin
    if Probe_mclk'event and Probe_mclk = '1' then        -- Process executed on each rising edge of Probe_mclk
        if counter0 = 1255 then
            OPERAT <= "010";                             -- Idle mode
        elsif change_to_acq = '1' then
            OPERAT <= "001";                             -- Acquisition mode
        end if;
    end if;
end process;

```

```

-- *****
--                               AQUISITION MODE OPERATION
-- *****

```

-- If isolated probe in acquisition mode, RAU\_clock is routed through the control logic to ADC\_clock.

```
ADC_Clock <= RAU_clock when OPERAT(0) = '1' else '0';
```

-- The ADC output enable line is enabled when system in acquisition mode.

```
ADC_OE <= '0' when OPERAT(0) = '1' else '1';
```

-- The fibre optic data line is disabled during idle mode, thus preventing the transmission of any data bytes.

```
FOTO <= '0';
```

-- Process that controls counter2. Counter2 is used when transmitting the ADC data to the parallel to serial encoders.

```

process(Probe_mclk)
begin
    if Probe_mclk'event and Probe_mclk = '1' then        -- Process executed on each rising edge of Probe_mclk
        if OPERAT(0) = '1' then                            -- Acquisition mode
            if (counter2 < 4) and (counter2 > 0) then      -- Increment counter2 from 0 to 3.
                counter2 <= counter2 + 1;
            elsif counter2 = 4 then
                if Flag = '0' and RAU_clock = '1' then    -- If the sample frequency is 5 MHz, it is possible..
                                                            -- that by the time counter2 is 4, the next sample..
                                                            -- clock cycle has begun. Counter2 is then set to..
                                                            -- 1 and not 0.
                    counter2 <= 1;
                else
                    counter2 <= 0;
                end if;
            elsif counter2 = 0 then
                if Flag = '0' and RAU_clock = '1' then    -- Start incrementing counter2 on a positive edge of..
                                                            -- RAU_clock.
                    counter2 <= 1;
                end if;
            end if;
        else
            counter2 <= 0;
        end if;
    end if;
end process;

```

```

        counter2 <= 0;                                -- Reset counter2 to zero during idle mode.
    end if;
end if;
end process;

-- Process that controls Flag. Flag is only used when the sampling frequency of the isolated probe is 5 MHz.
process(Probe_mclk)
begin
    if Probe_mclk'event and Probe_mclk = '1' then    -- Process executed on each rising edge of Probe_mclk
        if OPERAT(0) = '1' then                    -- Acquisition mode
            if RAU_clock = '0' then
                Flag <= '0';
            elsif counter2 = 3 then
                if RAU_clock = '1' then
                    Flag <= '1';
                end if;
            end if;
        else                                        -- idle mode
            Flag <= '0';
        end if;
    end if;
end process;

-- Process that controls SC_D, ENA and PS_Data. During acquisition mode, a valid data byte is present on..
-- ADC_Data 12.5 ns after each rising edge of ADC_clock. On the first rising edge of Probe_mclk after ADC_..
-- clock goes high (counter2 = 1), SC_D and ENA are pulsed low. The 8 MSBs of ADC_Data are also written..
-- to PS_Data. With the next rising edge of Probe_mclk (counter2 = 2) the 4 LSBs of ADC_Data are written..
-- to PS_Data together with 4 zero's. Thereafter ENA and SC_D are pulsed high again.
process(Probe_mclk)
begin
    if Probe_mclk'event and Probe_mclk = '1' then    -- Process executed on each rising edge of Probe_mclk
        if OPERAT(0) = '1' then                    -- Acquisition mode
            if counter2 = 1 then                    -- First + edge of Probe_mclk after + edge of ADC_clock
                SC_D <= '0';                        -- Data character to be sent
                ENA <= '0';                        -- Enable transmission
                PS_Data <= ADC_Data(11 downto 4);    -- 8 MSBs of ADC_Data written to PS_Data
            elsif counter2 = 2 then                 -- Second + edge of Probe_mclk after + edge of ADC_clock
                SC_D <= '0';                        -- Data character to be sent
                ENA <= '0';                        -- Enable transmission
                PS_Data <= ADC_Data(3 downto 0) & "0000"; -- 4 LSBs of ADC_Data with 4 zero's
            else
                SC_D <= '1';                        -- Special character to be sent
                ENA <= '1';                        -- Disable transmission
            end if;
        else
            SC_D <= '1';                            -- Special character to be sent
            ENA <= '1';                            -- Disable transmission
        end if;
    end if;
end process;

-- Disable Send Violation Symbol.
SVS <= '0';

-- Disable built-in self-test circuitry.
BISTEN <= '1';

-- Disable 'enable transmission on next clock edge'.
ENN <= '1';

end A;
```

**D2 FIFO/demux VHDL-code**

```

library IEEE;
use IEEE.std_logic_1164.all;

entity FIFO is
port( -- Control lines from RAU control module
    Reset          : in std_logic;           -- Reset input
    Px_OE          : in std_logic;           -- Enables data output onto RAU data bus
    Px_DRDY        : out std_logic;         -- Data ready indicator to RAU control module
    Px_Error       : out std_logic;         -- Error indicator to RAU control module

    -- IO pins to/from CY7B933
    SP_Data        : in std_logic_vector(7 downto 0); -- 8-Bit data from serial to parallel decoder
    Px_SC_D        : in std_logic;          -- Special/character data input
    Px_CKR         : in std_logic;          -- Receiving clock output of CY7B933

    -- RAU Data Bus
    RAUDataBus     : out std_logic_vector(15 downto 0)); -- RAU data bus
end FIFO;

architecture A of FIFO is
    signal InputBufferA : std_logic_vector(15 downto 0);
    signal InputBufferB : std_logic_vector(15 downto 0);
    signal OutputBuffer  : std_logic_vector(15 downto 0);
    signal in_counter    : integer range 3 downto 0;
    signal out_counter   : integer range 1 downto 0;
    signal sigError      : std_logic;
    signal EnableCounter : integer range 2 downto 0;
    signal EnOperation   : std_logic;
begin
    -- Process that controls EnableCounter and EnOperation. During idle mode the Reset input from the RAU control..
    -- module is driven high. It switches low during acquisition mode. Two data bytes are received consecu..
    -- tively from the CY7B933. The FIFO/demux waits until two consecutive data bytes have been rxed before..
    -- being enabled. EnableCounter keeps track of the rxed data bytes, and EnOperation enables operation of the..
    -- FIFO/demux.
    process(Px_CKR, Reset)
    begin
        if Reset = '1' then
            EnableCounter <= 0;
            EnOperation <= '0';
        elsif Px_CKR'event and Px_CKR = '1' then
            if EnableCounter < 2 then
                EnOperation <= '0';
                if Px_SC_D = '1' then
                    EnableCounter <= 0;
                elsif Px_SC_D = '0' then
                    EnableCounter <= EnableCounter + 1;
                end if;
            else
                EnOperation <= '1';
            end if;
        end if;
    end process;

    -- Process that controls in_counter and sigError. When a data byte is rxed from the CY7B933, the data is..
    -- written into either InputBufferA or InputBufferB and in_counter is incremented. When the data is..
    -- written over previous data that has not yet been read from the FIFO, an error has occurred and sigError..
    -- is pulsed high.

```

```

process(Px_CKR, Reset)
begin
  if Reset = '1' then
    in_counter <= 0;
    sigError <= '0';
  elseif Px_CKR'event and Px_CKR = '1' then
    if sigError = '0' and EnOperation = '1' then
      if Px_SC_D = '1' then
        if in_counter = 1 or in_counter = 3 then
          sigError <= '1';
        end if;
      elseif Px_SC_D = '0' then
        case in_counter is
          when 3 =>
            if out_counter = 0 then
              sigError <= '1';
            else
              in_counter <= 0;
            end if;
          when 2 =>
            in_counter <= 3;
          when 1 =>
            if out_counter = 1 then
              sigError <= '1';
            else
              in_counter <= 2;
            end if;
          when 0 =>
            in_counter <= 1;
        end case;
      end if;
    end if;
  end if;
end process;

```

-- Process that controls InputBufferA and InputBufferB. When a data byte is rxed from the CY7B933 it is..  
 -- written into either of these buffers, depending on the value of in\_counter.

```

process(Px_CKR, Reset)
begin
  if Reset = '1' then
    InputBufferA <= "0000000000000000";
    InputBufferB <= "0000000000000000";
  elseif Px_CKR'event and Px_CKR = '1' then
    if sigError = '0' and EnOperation = '1' then
      if Px_SC_D = '0' then
        case in_counter is
          when 0 =>
            InputBufferA(15 downto 8) <= SP_Data;
          when 1 =>
            InputBufferA(7 downto 0) <= SP_Data;
          when 2 =>
            InputBufferB(15 downto 8) <= SP_Data;
          when 3 =>
            InputBufferB(7 downto 0) <= SP_Data;
        end case;
      end if;
    end if;
  end if;
end process;

```

-- Process that controls out\_counter. Out\_counter is incremented each time a 16-bit word has been read..  
 -- from the FIFO/demux onto the RAU data bus. The positive edge of Px\_OE' is used for incrementing out\_...  
 -- counter.

```
process(Px_OE, Reset)
begin
  if Reset = '1' then
    out_counter <= 0; -- Initially reset out_counter.
  elsif Px_OE'event and Px_OE = '1' then -- Output-enable command was received.
    if sigError = '0' and EnOperation = '1' then -- No errors have occurred.
      if out_counter = 0 then -- Increment out_counter.
        out_counter <= 1;
      else
        out_counter <= 0;
      end if;
    end if;
  end if;
end process;
```

-- The data to be read next from the FIFO/demux is dependent on out\_counter.  
 OutputBuffer <= InputBufferA when out\_counter = 0 else InputBufferB when out\_counter = 1;

-- The Px\_OE signal controls the output of "OutputBuffer" to the 16-bit RAU data bus.  
 RAUDataBus <= OutputBuffer when Px\_OE = '0' else "ZZZZZZZZZZZZZZZZ";

-- If InputBufferA has been written to and this data has not been read out, out\_counter will be set to 0.  
 -- Px\_DRDY then driven high, indicating that this data still has to be read out. If InputBufferB's has not..  
 -- yet been read then out\_counter will be set to 1. In this case Px\_DRDY must also be pulsed high.

```
Px_DRDY <= '0' when EnOperation = '0'
  else '0' when Px_OE = '0'
  else '1' when (in_counter > 1 and out_counter = 0) or (in_counter < 2 and out_counter = 1)
  else '0';
```

-- A signal is assigned to Px\_Error so that the level of this output can be monitored.  
 Px\_Error <= sigError;

end A;

### D3 RAM interface unit (RIU) VHDL-code

```
library IEEE;
use IEEE.std_logic_1164.all;
```

entity RIU is

```
port( -- Control lines from RAU control module
  DIR          : in std_logic; -- Write/read to/from DRAM
  Latchclk     : in std_logic; -- Used to latch data into BufferA, B and C
  RIU_OE      : in std_logic; -- Output-enable (data onto RAU data bus)
  CS          : in std_logic; -- Chip select (between RIU0 and RIU1)
  cycle_counter : in std_logic;

  -- RAU Data Bus
  RAUDataBus   : inout std_logic_vector(15 downto 0);

  -- 32-Bit bus to RAM
  RAMDataBus   : inout std_logic_vector(31 downto 0);

  -- Debugging and other interface lines
  reset        : in std_logic;
  SGND         : in std_logic;
```



**D4 DRAM controller VHDL-code**

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity RAMCONTR is
port( -- Master Clock (50 MHz)
      mclk          : in std_logic;

      -- Control lines from RAU control module
      RD            : in std_logic;          -- read
      WR            : in std_logic;          -- write

      -- Outputs to Dynamic RAM on RAM control bus
      WE            : out std_logic;         -- write enable
      RASL          : out std_logic;         -- row address strobe
      RASH          : out std_logic;         -- row address strobe
      CASL0         : out std_logic;         -- column address strobe
      CASH0         : out std_logic;         -- column address strobe
      CASL1         : out std_logic;
      CASH1         : out std_logic;
      simm_addr     : out std_logic_vector(10 downto 0); -- SIMM address (row/column address)

      -- Address Input from Address Generator
      ram_addr      : in std_logic_vector(22 downto 0); -- RAM address

      -- Jumper settings
      present       : in std_logic;
      size          : in std_logic_vector(1 downto 0);
      auto          : in std_logic;
      PD            : in std_logic_vector(1 downto 0); -- PD1(1) and PD2(0) from SIMMs

      -- Extra
      mreset        : in std_logic;
      width         : in std_logic;
      ram_addr23    : in std_logic;
      extra1        : out std_logic;
      extra2        : out std_logic;
      extra3        : out std_logic);
end RAMCONTR;

architecture A of RAMCONTR is
  signal bitpipe      : std_logic_vector(5 downto 0); -- used for timing during read/write/reset cycles
  signal reset_pipe   : std_logic;                  -- used to reset bitpipe
  signal request      : std_logic;                  -- indicates refresh/read/write request
  signal start_pipe   : std_logic;                  -- starts bitpipe
  signal source0      : std_logic;                  -- assigned to bitpipe(0) on each clock edge
  signal source5      : std_logic;                  -- assigned to bitpipe(5) on each clock edge
  signal refr_addr    : std_logic_vector(10 downto 0); -- contains next row address to be refreshed
  signal refr_timer   : std_logic_vector(5 downto 0); -- keeps track of when next refresh is necessary
  signal refr_req     : std_logic;                  -- indicates that a refresh should be executed
  signal RDorWR       : std_logic;                  -- low when RD or write is low
  signal data_req     : std_logic;                  -- launches a data request
  signal data_op      : std_logic;                  -- indicates that a data request is being executed
  signal wr_latched  : std_logic;                  -- indicates when a 1-0 transition detected on WR
  signal simm_specs   : std_logic_vector(2 downto 0); -- encapsulates SIMM sizes and number of SIMMs
  signal sixt_meg     : std_logic;                  -- HI when 16-MByte SIMMs are implemented
  signal row_addr     : std_logic_vector(10 downto 0); -- contains a row address for DRAM

```

```

signal col_addr      : std_logic_vector(10 downto 0); -- contains a column address for DRAM
signal addr_select  : std_logic_vector(1 downto 0); -- contains refresh address for DRAM
signal RAS          : std_logic;                   -- indicates when one of RASL or RASH must be low
signal RAS_decoding : std_logic_vector(2 downto 0); -- used for distinguishing between RASL and RASH
signal RASL_select  : std_logic;                   -- indicates RASL is selected
signal RASH_select  : std_logic;                   -- indicates RASH is selected
signal CAS          : std_logic;                   -- indicates when one of CAS0 or CAS1 must be low
signal CAS0_select  : std_logic;                   -- indicates CAS0 is selected
signal CAS1_select  : std_logic;                   -- indicates CAS1 is selected
begin

extra1 <= '0';
extra2 <= '0';
extra3 <= '0';

-- *****
-- BITPIPE CONTROL (FOR DATA OPERATION AND REFRESH CYCLES)
-- *****

-- The RAM Controller controls both the refreshing of the RAM, and the writing and reading of data..
-- to/from memory. For a refresh/read/write the RAS line of the relevant RAM block must be pulsed..
-- low for 80ns, and for a read/write command the CAS line must also be pulsed low for the last 40ns..
-- that RAS is low. This timing is established by using a 6-bit bitpipe that is started by either..
-- a refresh request or a data request (read or write command).

-- When initialised, the LSB of the bitpipe is set to '1'. With every clock cycle thereafter (for..
-- 4 clock cycles the bit to the left of the '1' is also set to '1'. Thus after 2 clock cycles the..
-- bitpipe will be 000111, and after 4 it will be 011111. The bitpipe is cycled through once for..
-- every refresh cycle and once for every read or write cycle.

-- The MSB of the bitpipe is used to reset the bitpipe. During a refresh cycle the bitpipe takes 6..
-- clock cycles before it is reset. During a read or write cycle the MSB of the bitpipe is not driven..
-- high before the RD or WR input is deactivated. When the most significant bit switches to '1', the..
-- bitpipe is reset, making it 000000 again.

-- Thus, when a refresh needs to be done, or when one of the RD or WR inputs are activated, the..
-- bitpipe is started and cycled through once. If a RD or WR command is received and the controller..
-- is in a refresh cycle, the refresh is completed before the RD or WR command is executed. If the..
-- command is received and no refresh is currently in progress, a refresh is first done before the..
-- RD or WR command is executed.

-- The bitpipe is mreset when the MSB of the pipe is high (1)
process(mreset, mclk)
begin
    if mreset = '1' then
        reset_pipe <= '1';
    elsif mclk'event and mclk = '1' then
        reset_pipe <= bitpipe(5);
    end if;
end process;

-- Process that controls the bitpipe
process(reset_pipe, mclk)
begin
    if reset_pipe = '1' then
        bitpipe <= "000000";
    elsif mclk'event and mclk = '1' then
        bitpipe(5) <= source5;
        bitpipe(4) <= bitpipe(3);
        bitpipe(3) <= bitpipe(2);
        bitpipe(2) <= bitpipe(1);
    end if;
end process;

```

```

        bitpipe(1) <= bitpipe(0);
        bitpipe(0) <= source0;
    end if;
end process;

-- *****
--      INITIALISATION OF A BITPIPE CYCLE (FOR REFRESH OR RD/WR)
-- *****

-- The bitpipe is initialised by a refresh or data request (discussed later). If one of these are..
-- high, "request" switches high, causing "start_pipe" to pulse high, and the bitpipe cycle is started.

request <= '1' when refr_req = '1' or data_req = '1' else '0';

-- Once a refresh or data request has been received, the bitpipe must be started. If the bitpipe is..
-- not zero when the request is encountered, a refresh cycle is currently in progress. In such a case..
-- "request" will remain high ("data_req" is high until the data operation is performed) meaning that..
-- "start_pipe" remains high. When the refresh cycle is complete the bitpipe is reset, and the data..
-- operation is attended to. The following assignments then take place in this order:
--     "data_op" pulses high,
--     "bitpipe0" pulses high, starting bitpipe cycle,
--     "data_req" returns to zero,
--     and "start_pipe" returns to zero.

start_pipe <= '1' when request = '1' and reset_pipe = '0' else '0';
source0 <= '1' when start_pipe = '1' or bitpipe(0) = '1' else '0';

-- If there is no data operation in progress, the bitpipe must complete its cycle in six clock cycles..
-- (bitpipe5 gets the previous bitpipe4 value with every clock cycle). If a data operation is in progress..
-- then only after the RD line has returned to its inactive high level, does bitpipe5 get bitpipe4's..
-- value ('1' at that time), and can the bitpipe reset. This ensures that the read cycle is completed..
-- before the bitpipe is reset.

source5 <= bitpipe(4) when data_op = '0' or RD = '1' else '0';

-- *****
--                      REFRESH REQUESTS
-- *****

-- Each address in the memory bank has to be refreshed every 16ms. Thus in the worst case scenario..
-- (16MByte SIMMs are used) a refresh needs to be done every 15.625us (16ms/1024). In this controller..
-- a refresh is done every 12.8us (after 64 clock cycles). Every 64 clock cycles a refresh is requested..
-- (refr_req is pulsed high). When the refresh is executed, the bitpipe is cycled through and the..
-- refresh address is incremented (next address to be reset). If data_op is high when the bitpipe..
-- is complete, it means that a RD or WR cycle is just complete. In this case the refresh address..
-- must not be incremented. A refresh timer indicates when a refresh must take place. Each time the..
-- bitpipe is reset, the refresh timer is reset to zero, and then incremented on every clock edge..
-- thereafter. When it reaches 111111 (with a 50 MHz clock this is after 1.28us), a refresh is requested.

process(reset_pipe)
begin
    wait until reset_pipe = '1';
    if data_op = '0' then
        refr_addr <= refr_addr + 1;
    end if;
end process;

process(reset_pipe, mclk)
begin
    if reset_pipe = '1' then
        refr_timer <= "000000";
    end if;
end process;

```

```

        elsif mclk'event and mclk = '1' then
            refr_timer <= refr_timer + 1;           -- increment refresh timer
        end if;
    end process;

refr_req <= '1' when refr_timer = "111111" else '0'; -- after 1.28us

-- *****
--          DATA REQUESTS (READ OR WRITE COMMANDS)
-- *****

-- If RD or WR is zero then a data operation must take place, or is taking place. The signal "RDorWR"..
-- is low if either the RD or WR inputs are activated (pulsed low). As soon as a 1-0 transition is..
-- detected on RDorWR, a data operation is requested (a read or write must be performed). This 1-0..
-- transition will set "data_req" to '1'. A refresh is then done immediately, whereafter the data..
-- operation is attended to (data_op = '1') and "data_req" returns to zero.

-- A write command is given by a 1-0 transition on the WR input. If this transition is encountered,..
-- "wr_latched" is set high. As soon as the write command is attended to (after a refresh), the WE..
-- output to the RAM is activated until the relevant data has been written to the RAM (in other words..
-- when the bitpipe is reset).

RDorWR <= RD and WR;

process(data_op, bitpipe(1), RDorWR)
begin
    if (data_op = '1' and bitpipe(1) = '1') then
        data_req <= '0';           -- Data request has been attended to.
    elsif RDorWR'event and RDorWR = '0' then
        data_req <= '1';         -- Clear request flag.
    end if;                       -- Read or write command detected.
    end process;                 -- Launch data request.

process(reset_pipe)
begin
    wait until reset_pipe = '0';
    data_op <= data_req;
end process;

process(WR, reset_pipe)
begin
    if (reset_pipe = '1' and data_op = '1') then
        wr_latched <= '0';
    elsif WR'event and WR = '0' then
        wr_latched <= '1';       -- WR command
    end if;
end process;

-- The write enable (WE) output is activated two clock cycles (40 ns) after a write operation has started.
WE <= '0' when (bitpipe(2) = '1' and data_op = '1' and wr_latched = '1') else '1';

-- *****
--          CONFIGURATION FOR SPECIFICATIONS FOR SIMMS
-- *****

-- The specifications for the SIMMS that are used are encapsulated in the 3-bit signal "simm_specs".
-- This signal is interpreted as follows:
-- bits 1 - 0 => "00" - 4 MByte SIMM(s)
--              "01" - 8 MByte SIMM(s)
--              "10" - 16 MByte SIMM(s)
-- bit 2      => '0' - 1 SIMM

```

```

--          '1' - 2 SIMMS
-- These specifications can be detected automatically (from PD outputs of SIMMs), or set manually..
-- with "size"-jumpers.

process(PD, auto, size)
begin
    if auto = '1' then
        simm_specs(0) <= size(0);
        simm_specs(1) <= size(1);
    else
        case PD is
            when "11" =>
                simm_specs(1 downto 0) <= "00"; -- 4 MByte SIMMS
            when "00" =>
                simm_specs(1 downto 0) <= "01"; -- 8 MByte SIMMS
            when "10" =>
                simm_specs(1 downto 0) <= "10"; -- 16 MByte SIMMS
            when others =>
        end case;
    end if;
end process;

simm_specs(2) <= present; -- 1/2 SIMMS present

-- *****
-- ADDRESS DECODING
-- *****

-- If 16M SIMMS are used, there are 11 address line outputs to the RAM. Of the 23 address lines,..
-- bits 0-10 are used for the column address, and bits 11-21 are used for the row address. With 4/8M..
-- SIMMS there are only 10 address lines per SIMM. Bits 0-9 are then used as the column address, and..
-- bits 10-19 are used for the row address. The remaining address lines are used for the RAS and CAS..
-- decoding and are discussed later.

process(simm_specs)
begin
    if simm_specs(1 downto 0) = "10" then
        sixt_meg <= '1'; -- 16 MByte SIMMS
    else
        sixt_meg <= '0'; -- 4/8 MByte SIMMS
    end if;
end process;

row_addr <= '0' & ram_addr(9 downto 0) when sixt_meg = '0' else ram_addr(10 downto 0);
col_addr <= '0' & ram_addr(19 downto 10) when sixt_meg = '0' else ram_addr(21 downto 11);

-- "Addr_select" determines whether a refresh address or the row or column address must be..
-- output to the RAM. Bit1 of "addr_select" is assigned the value of "data_op". If bit1 is 0,..
-- no data operation is in progress, thus a refresh address must be output. If bit1 is 1, a..
-- data operation is in progress, thus a row or column address must be output. During a data..
-- operation the row address is first put out to the RAM and then the column address. Bit0 of..
-- "addr_select" is assigned the value of bit-pipe(2). Before bitpipe(2) is '1' the row address..
-- is output, and if bitpipe(2) is '1' the column address is output. Therefore "addr_select" is..
-- interpreted as follows:
-- "0X" - No data operation; refresh address to be output.
-- "10" - Data operation; row address to be output.
-- "11" - Data operation; column address to be output.

addr_select <= data_op & bitpipe(2);

simm_addr <= refr_addr when addr_select = "00"

```

```

else refr_addr when addr_select = "01"
else row_addr when addr_select = "10"
else col_addr when addr_select = "11";

```

```

-- *****
--                                     RAS DECODING
-- *****

```

```

-- It is accepted that when reading from or writing to the memory the 23-bit address input to the RAM..
-- controller will start at 0 and increment to 111...111 (independent on the size of the SIMMS). The..
-- RAS decoding involves the activation of RASL and RASH during read/write and refresh cycles. The..
-- selection of RASL and RASH is interpreted as follows:
-- 4/16 MByte SIMMs : Use only RASL.
-- 8 MByte SIMMs   : RASL - Side0; RASH - Side1.

-- RAS_decoding indicates whether RASL or RASH are to be pulsed for a refresh/read/write cycle. If
-- RAS_decoding(0) is high during a data cycle, RASL must be pulsed. If RAS_decoding(1) is high..
-- during a data cycle, RASH must be pulsed. During refresh cycles RASL and RASH are pulsed. RAS_..
-- decoding(2) indicates when a refresh cycle is to take place.

```

```
process(simm_specs)
```

```
begin
```

```

    if simm_specs(1 downto 0) = "01" then
        RAS_decoding(0) <= ram_addr(21);
    else
        RAS_decoding(0) <= '1';
    end if;

```

```
end process;
```

```
process(simm_specs)
```

```
begin
```

```

    if simm_specs(1 downto 0) = "01" then
        RAS_decoding(1) <= not(ram_addr(21));
    else
        RAS_decoding(1) <= '0';
    end if;

```

```
end process;
```

```
RAS_decoding(2) <= not(data_op);
```

```

-- RASL is selected when RAS_decoding(0) is high or during a refresh. RASH is selected when RAS_..
-- decoding(1) is high or during a refresh.

```

```
RASL_select <= '1' when (RAS_decoding(2) = '1' or RAS_decoding(0) = '1') else '0';
```

```
RASH_select <= '1' when (RAS_decoding(2) = '1' or RAS_decoding(1) = '1') else '0';
```

```

-- For a refresh/read/write operation the RAS line of the RAM block being refreshed/read from/..
-- written to, has to be low for at least 80ns. This is accomplished by using the bitpipe. Once..
-- the bitpipe is started, it will take 4 clock cycles (80ns) from the time that bitpipe(1) is..
-- high to when bitpipe(5) is high. The signal "RAS" is high during this time and indicates that..
-- either one of the RAS lines (RASH or RASL) must be pulsed low.

```

```
RAS <= '1' when bitpipe(1) = '1' and bitpipe(5) = '0' else '0';
```

```
RASL <= '0' when (RAS = '1' and RASL_select = '1') else '1';
```

```
RASH <= '0' when (RAS = '1' and RASH_select = '1') else '1';
```

```

-- *****
--                                     CAS DECODING
-- *****

```

```

-- CAS decoding involves the selection of the correct CAS line when reading from or writing to the..
-- DRAM. CAS0 is the CAS line for SIMM0 and CAS1 is the CAS line for SIMM1. With 4/8 Mbyte SIMMs..
-- ram_addr(0..19) are used as the row and column addresses for the DRAM SIMMs. Ram_addr(20) then..

```

```
-- selects between SIMM0 and SIMM1. For the 8 Mbyte SIMMs ram_addr(21) selects between the DRAM..
-- units on side0 and side1 of the SIMMs. 16 MByte SIMMs use ram_addr(0..21) for the row and column..
-- addresses of the SIMMs. Ram_addr(22) is used to distinguish between SIMM0 and SIMM1 for 16 Mbyte..
-- SIMMs.
```

```
process(simm_specs)
begin
  if simm_specs(1 downto 0) = "00" then           -- 4 MByte SIMMs
    CAS0_select <= not(ram_addr(20));
  elsif simm_specs(1 downto 0) = "01" then       -- 8 MByte SIMMs
    CAS0_select <= not(ram_addr(20));
  else                                           -- 16 MByte SIMMs
    CAS0_select <= not(ram_addr(22));
  end if;
end process;
```

```
process(simm_specs)
begin
  if simm_specs(1 downto 0) = "00" then         -- 4 MByte SIMMs
    CAS1_select <= ram_addr(20);
  elsif simm_specs(1 downto 0) = "01" then     -- 8 MByte SIMMs
    CAS1_select <= ram_addr(20);
  else                                           -- 16 MByte SIMMs
    CAS1_select <= ram_addr(22);
  end if;
end process;
```

```
-- CAS must be low for at least 40ns for a read or write cycle. It takes 40 ns from the time that..
-- bitpipe(3) goes high to the time that bitpipe(5) goes high. CAS must also only be enabled during..
-- a data operation (data_op = '1'), and not during a refresh cycle (data_op = '0'). If these..
-- conditions are true the "CAS" signal is set to '1'.
CAS <= '1' when (bitpipe(3) = '1' and bitpipe(5) = '0' and data_op = '1') else '0';
```

```
-- Which of the CAS-lines are active when CAS is high is determined by the CAS_select signals.
```

```
CASL0 <= '0' when CAS = '1' and CAS0_select = '1' else '1';
CASH0 <= '0' when CAS = '1' and CAS0_select = '1' else '1';
CASL1 <= '0' when CAS = '1' and CAS1_select = '1' else '1';
CASH1 <= '0' when CAS = '1' and CAS1_select = '1' else '1';
```

```
end A;
```

## D5 DRAM address generator VHDL-code

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

```
entity ADDRGEN is
```

```
port( -- Master clock signal
      mclk      : in std_logic;           -- from 20 MHz crystal oscillator

      -- Interface lines from microcontroller
      uC_bus    : in std_logic_vector(7 downto 0); -- Microcontroller data bus
      ALE       : in std_logic;           -- address latch enable
      uC_WR     : in std_logic;           -- write line from microcontroller

      -- Control line from RAU control module
      INC       : in std_logic;           -- increment address
```

```

-- Output address to DRAM controller
ram_addr : out std_logic_vector(22 downto 0);           -- 23-bit DRAM address

-- extra
mclkout  : out std_logic;
mreset   : in  std_logic;
RD       : in  std_logic;
TRIG     : in  std_logic;
RA23     : out std_logic;
DIP3     : out std_logic;
DIP4     : out std_logic;
DIP5     : out std_logic;
DIP6     : out std_logic;
end ADDRGEN;

architecture A of ADDRGEN is
    signal config_reg      : std_logic_vector(7 downto 0); -- configuration register
    signal access_reg      : std_logic_vector(7 downto 0); -- access control register
    signal ALE_buf         : std_logic_vector(1 downto 0); -- used to detect positive edge of ALE
    signal dev_sel         : std_logic;                   -- indicates when this device is selected
    signal Enable          : std_logic;                   -- enable incrementing of RAM address
    signal Reset           : std_logic;                   -- reset RAM address
    signal ad1, ad2        : std_logic_vector(7 downto 0);
    signal ad3             : std_logic_vector(6 downto 0); -- ad1, ad2 & ad3 form the generated DRAM address
begin
    *****
    --
    -- CONFIGURATION OF RAM ADDRESS GENERATOR
    --
    *****

    -- With every clock edge the level of ALE is shifted into ALE_buf. When ALE is "01" it means that..
    -- a 0-1 transition has taken place on ALE. The data on the microcontroller bus is then latched..
    -- into the access control register.
    process(mclk)
    begin
        if mclk'event and mclk = '1' then
            ALE_buf(0) <= ALE;
            ALE_buf(1) <= ALE_buf(0);           -- shift ALE level into ALE_buf
        end if;
    end process;

    process(mclk)
    begin
        if mclk'event and mclk = '1' then
            if ALE_buf = "01" then             -- positive edge detected
                access_reg <= uC_bus;
            end if;
        end if;
    end process;

    -- If the access control register contains "10100000" this device is selected by the microcontroller.
    -- The data on the microcontroller bus is then latched into the configuration register on the rising..
    -- edges of uC_WR.
    dev_sel <= '1' when access_reg = "10100000" else '0';

    process(uC_WR)
    begin
        if uC_WR'event and uC_WR = '1' then   -- positive edge of uC_WR
            if dev_sel = '1' then             -- device selected
                config_reg <= uC_bus;
            end if;
        end if;
    end process;
end architecture A;

```

```

    end if;
end process;

-- The two least significant bits of the configuration register respectively reset the RAM address..
-- and enable incrementing of the RAM address.
Reset <= config_reg(0) when dev_sel = '1' else '0';
Enable <= config_reg(1) when dev_sel = '1' else '0';

--*****
--                               CONTROL OF DRAM ADDRESS
--*****

-- Ad1, ad2 and ad3 form the 23-bit address that is output to the DRAM module. This address is..
-- incremented on each rising edge of INC (if Enable is high).
process(Reset, INC)
begin
    if Reset = '1' then
        ad1 <= "00000000";
        ad2 <= "00000000";
        ad3 <= "00000000";
    elsif INC'event and INC = '1' then
        if Enable = '1' then
            -- positive edge of INC
            -- incrementing enabled
            ad1 <= ad1 + 1;
            if ad1 = "11111111" then
                -- |
                ad2 <= ad2 + 1;
                if ad2 = "11111111" then
                    -- | Increment 23-bit DRAM address
                    ad3 <= ad3 + 1;
                    -- |
                end if;
            end if;
        end if;
    end if;
end process;

ram_addr <= ad3 & ad2 & ad1;

DIP3 <= ad1(0);
DIP4 <= ad1(1);
DIP5 <= ad1(2);
DIP6 <= ad1(3);
mclkout <= RD;
RA23 <= '0';

end A;

```

## D6 RAU control module VHDL-code

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity CONTROLF is
port( -- Master Clock
    mclk          : in std_logic;
    mreset        : in std_logic;
                -- External reset for debugging purposes

    -- Interface to two FIFO/demuxes
    P0_DRDY       : in std_logic;
    P0_OE         : out std_logic;

```

```

P0_Error      : in std_logic;
P1_DRDY      : in std_logic;
P1_OE        : out std_logic;
P1_Error      : in std_logic;
Reset         : out std_logic;

-- Interface to two RIUs
DIR           : out std_logic;
RIU_OE        : out std_logic;
Latchclk      : out std_logic;
cycle_counter : out std_logic;
CS            : out std_logic;

-- Configuration and Control Bits to/from Microcontroller
uC_bus        : in std_logic_vector(7 downto 0);
uC_WR         : in std_logic;
ALE           : in std_logic;

-- Interface to Fibre Optic Transmit Logic
P0_Clock      : out std_logic;
P1_Clock      : out std_logic;

-- Interface to RAM Address Generators
INC           : out std_logic;

-- Control to RAM Controllers
WR            : out std_logic;
RD            : out std_logic;

-- Extra lines
BISTEN1       : out std_logic;
RVS1, RVS0    : in std_logic;
ECP_ACK       : in std_logic;
ECP_PACK      : out std_logic;
ECP_CLK       : out std_logic;
reset_ram     : out std_logic;
BISTEN0       : out std_logic;
TRIG          : out std_logic;
up_RD         : in std_logic);
end CONTROLF;

architecture A of CONTROLF is
    signal access_reg      : std_logic_vector(7 downto 0);
    signal op_mode         : std_logic_vector(7 downto 0);
    signal op_specs        : std_logic_vector(7 downto 0);
    signal opmode_sel      : std_logic;
    signal opspects_sel    : std_logic;
    signal ALE_buf         : std_logic_vector(1 downto 0);
    signal WR_buf          : std_logic_vector(1 downto 0);
    signal One100ns_counter : integer range 4 downto 0;
    signal One100ns_pulse  : std_logic;
    signal FiveMHzClk      : std_logic;
    signal Five100ns_counter : integer range 24 downto 0;
    signal Five100ns_pulse : std_logic;
    signal OneMHzClk       : std_logic;
    signal Five100kHzClk   : std_logic;
    signal FiveMicroSec_counter : integer range 4 downto 0;
    signal FiveMicroSec_pulse : std_logic;
    signal One100kHzClk    : std_logic;
    signal FiftykHzClk     : std_logic;
    signal FiftyMicroSec_counter : integer range 4 downto 0;

```

```

signal FiftyMicroSec_pulse : std_logic;
signal TenHzClk           : std_logic;
signal sample_freq       : std_logic_vector(2 downto 0);
signal sample_sig       : std_logic;
signal two_channels_active : std_logic;
signal channel0_active  : std_logic;
signal channel1_active  : std_logic;
signal setup_timer      : integer range 511 downto 0;
signal setup_time_elapsed : std_logic;
signal setup_timer_eq_510 : std_logic;
signal P0_valid_data    : std_logic;
signal P1_valid_data    : std_logic;
signal Enable           : std_logic;
signal toggle           : std_logic;
signal handle           : std_logic;
signal OE0              : std_logic;
signal OE1              : std_logic;
signal one_ramblock_active : std_logic;
signal latch_acq        : std_logic;
signal cycle_acq        : std_logic;
signal CS_counter       : integer range 3 downto 0;
signal CS_acq           : std_logic;
signal dwnl_timer       : integer range 99 downto 0;
signal latch_dwnl       : std_logic;
signal cycle_dwnl       : std_logic;
signal CS_dwnl          : std_logic;
signal RIU_OE_dwnl      : std_logic;
signal OE_counter       : integer range 1 downto 0;
signal INC_acq          : std_logic;
signal INC_dwnl         : std_logic;
begin
-- *****
--          CONFIGURATION OF CONTROL FPGA FROM MICROCONTROLLER
-- *****

-- On the first positive edge of mclk after a 0-1 transition on ALE, the data on uC_bus is latched..
-- into the access control register in the RAU control module.
process(mreset, mclk)
begin
    if mreset = '1' then
        ALE_buf <= "00";
    elsif mclk'event and mclk = '1' then
        ALE_buf(1) <= ALE_buf(0);
        ALE_buf(0) <= ALE;                    -- shift ALE level into ALE_buf
    end if;
end process;

process(mreset, mclk)
begin
    if mreset = '1' then
        access_reg <= "00000000";
    elsif mclk'event and mclk = '1' then
        if ALE_buf = "01" then                -- rising edge of ALE detected
            access_reg <= uC_bus;            -- latch uC_data into access control register
        end if;
    end if;
end process;

```

```

-- The op_mode and op_specs registers are selected when the access control register contains..
-- "00001010" or "00001111" respectively.
opmode_sel <= '1' when access_reg = "00001010" else '0';
opspecs_sel <= '1' when access_reg = "00001111" else '0';

-- If either of the configuration registers (op_mode/op_specs) are selected, the data on uC_bus..
-- must be latched into that register on the positive edges of WR.
process(mclk, mreset)
begin
    if mreset = '1' then
        WR_buf <= "00";
    elsif mclk'event and mclk = '1' then
        WR_buf(1) <= WR_buf(0);
        WR_buf(0) <= uC_WR;
        -- shift uC_WR level into WR_buf
    end if;
end process;

process(mreset, mclk)
begin
    if mreset = '1' then
        op_mode <= "00000000";
    elsif mclk'event and mclk = '1' then
        if WR_buf = "01" and opmode_sel = '1' then
            op_mode <= uC_bus;
        end if;
    end if;
end process;

process(mreset, mclk)
begin
    if mreset = '1' then
        op_specs <= "00000000";
    elsif mclk'event and mclk = '1' then
        if WR_buf = "01" and opspecs_sel = '1' then
            op_specs <= uC_bus;
        end if;
    end if;
end process;

-- *****
-- GENERATION OF SAMPLE CLOCKS FOR ISOLATED PROBES
-- *****

-- The possible sampling frequencies to be sent to the isolated probes are 5MHz, 1MHz, 500kHz, 100kHz,
-- 50kHz and 10kHz. Signals of these frequencies are generated by using mclk and several counters.

-- One100ns_counter is incremented with every rising edge of mclk until it reaches 4. It then starts..
-- at zero again. With a 50 MHz clock this counter is cycled from 0 to 4 within every 100 ns. When it..
-- reaches 4, One100ns_pulse is pulsed high for 20 ns. This pulse is in turn used to toggle FiveMHzCik..
-- in order to produce a 50% duty cycle signal with frequency 5MHz.
process(mclk)
begin
    if mclk'event and mclk = '1' then
        if One100ns_counter = 4 then
            One100ns_counter <= 0;
        else
            One100ns_counter <= One100ns_counter + 1;
        end if;
    end if;
end process;

```

```
One100ns_pulse <= '1' when One100ns_counter = 4 else '0';
```

```
process(One100ns_pulse)
begin
  if One100ns_pulse'event and One100ns_pulse = '1' then
    FiveMHzClk <= not(FiveMHzClk);
  end if;
end process;
```

-- Five100ns\_counter is incremented with every rising edge of mclk until it reaches 24. It then starts..  
 -- at zero again. With a 50 MHz clock this counter is cycled from 0 to 4 within every 500 ns. When it..  
 -- reaches 24, Five100ns\_pulse is pulsed high for 20 ns. This pulse is in turn used to toggle OneMHzClk..  
 -- in order to produce a 50% duty cycle signal with frequency 1MHz.

```
process(mclk)
begin
  if mclk'event and mclk = '1' then
    if Five100ns_counter = 24 then
      Five100ns_counter <= 0;
    else
      Five100ns_counter <= Five100ns_counter + 1;
    end if;
  end if;
end process;
```

```
Five100ns_pulse <= '1' when Five100ns_counter = 24 else '0';
```

```
process(Five100ns_pulse)
begin
  if Five100ns_pulse'event and Five100ns_pulse = '1' then
    OneMHzClk <= not(OneMHzClk);
  end if;
end process;
```

-- In order to generate a signal with frequency 500kHz, the already generated OneMHzClk signal is..  
 -- used. On every positive edge of OneMHzClk, the Five100kHz signal is toggled. OneMHzClk is also..  
 -- used to generate a 100kHz signal. FiveMicroSec\_counter is incremented on each rising edge of..  
 -- OneMHzClk to 4, and then starts at zero again. FiveMicroSec\_pulsed is pulsed high each time..  
 -- FiveMicroSec\_counter reaches 4. This signal is then used to generate the 100 kHz signal.

```
process(OneMHzClk)
begin
  if OneMHzClk'event and OneMHzClk = '1' then
    Five100kHzClk <= not(Five100kHzClk);
    if FiveMicroSec_counter = 4 then
      FiveMicroSec_counter <= 0;
    else
      FiveMicroSec_counter <= FiveMicroSec_counter + 1;
    end if;
  end if;
end process;
```

```
FiveMicroSec_pulse <= '1' when FiveMicroSec_counter = 4 else '0';
```

```
process(FiveMicroSec_pulse)
begin
  if FiveMicroSec_pulse'event and FiveMicroSec_pulse = '1' then
    One100kHzClk <= not(One100kHzClk);
  end if;
end process;
```

```

-- In similar fashion the One100kHzClk-signal is used to generate a 50 kHz and 10 kHz clock signal.
process(One100kHzClk)
begin
    if One100kHzClk'event and One100kHzClk = '1' then
        FiftyHzClk <= not(FiftyHzClk);
        if FiftyMicroSec_counter = 4 then
            FiftyMicroSec_counter <= 0;
        else
            FiftyMicroSec_counter <= FiftyMicroSec_counter + 1;
        end if;
    end if;
end process;

FiftyMicroSec_pulse <= '1' when FiftyMicroSec_counter = 4 else '0';

process(FiftyMicroSec_pulse)
begin
    if FiftyMicroSec_pulse'event and FiftyMicroSec_pulse = '1' then
        TenkHzClk <= not(TenkHzClk);
    end if;
end process;

-- The three least significant bits of the op_specs register contains information on the sample..
-- frequency required by the isolated probes. These bits determine which of the sampling frequency..
-- signals are transmitted to the isolated probes during acquisition mode.
sample_freq <= op_specs(2 downto 0);

with sample_freq select
    sample_sig <= TenkHzClk when "000",
                 FiftyHzClk when "001",
                 One100kHzClk when "010",
                 Five100kHzClk when "011",
                 OneMHzClk when "100",
                 FiveMHzClk when "101",
                 '0' when others;

-- The sampling frequency signal is sent to the active isolated probes during acquisition mode.
-- The current operating mode of the system is determined by op_mode(0). The isolated probes that..
-- are activated depends on op_specs(7) and op_specs(6).
P0_Clock <= sample_sig when (op_mode(0) = '1' and op_specs(7) = '1') else '0';
P1_Clock <= sample_sig when (op_mode(0) = '1' and op_specs(6) = '1') else '0';

-- *****
--                               CONTROL OF FIFO/DEMUXES
-- *****

-- As soon as the RAU control module switches to acquisition mode, a setup time is entered to allow..
-- all the system components to react to the mode change. In this time the FIFO/demuxes are reset,..
-- causing the FIFO/demuxes to discard any incoming data from the CY7B933s. This ensures that by..
-- the time data is being written into memory, it is data which was sampled by the ADC.
process(mclk)
begin
    if mclk'event and mclk = '1' then
        mreset          : in std_logic;          -- External reset for debugging purposes

        if op_mode(0) = '0' then                -- not in acquisition mode
            setup_timer <= 0;                   -- reset setup_timer to zero
        else                                     -- during acquisition mode
            if setuptime_elapsed = '0' and setuptimer_eq_510 = '0' then
                setup_timer <= setup_timer + 1; -- increment setup_timer to value 511, then stop
            end if;
        end if;
    end if;
end process;

```

```

        end if;
    end if;
end process;

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if setup_timer = 511 then
            setuptime_elapsed <= '1';           -- setup time has elapsed
        else
            setuptime_elapsed <= '0';
        end if;
    end if;
end process;

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if setup_timer = 510 then
            setuptimer_eq_510 <= '1';           -- one clock cycle before setup time elapsed
        else
            setuptimer_eq_510 <= '0';
        end if;
    end if;
end process;

-- FIFO/demuxes are deactivated when not in acquisition mode. During acquisition mode, these..
-- devices are deactivated during the initial setup time.
Reset <= '1' when (op_mode(0) = '0' or setuptime_elapsed = '0' or mreset = '1') else '0';

-- The FIFOs/demuxes receive data from the isolated probes which has to be transferred to the RAU..
-- data bus. When a FIFO/demux contains data to be transferred, its DRDY line is pulsed high. The..
-- RAU control module monitors when either of the DRDY lines pulse high, and in turn controls the..
-- OE lines to the FIFO/demuxes, and the data is transferred to the RAU data bus.

-- Two_channels_active, channel0_active and channel1_active are signals that indicate which of the
-- sampling channels have been activated. This information is encapsulated in the op_specs register.
process(op_specs)
begin
    if op_specs(7 downto 6) = "11" then         -- channel 0 & channel 1 active
        two_channels_active <= '1';
    else
        two_channels_active <= '0';
    end if;
end process;

process(op_specs)
begin
    if op_specs(7 downto 6) = "10" then         -- only channel 0 active
        channel0_active <= '1';
    else
        channel0_active <= '0';
    end if;
end process;

process(op_specs)
begin
    if op_specs(7 downto 6) = "01" then         -- only channel 1 active
        channel1_active <= '1';
    else
        channel1_active <= '0';
    end if;
end process;

```

```

    end if;
end process;

```

-- The DRDY signals for channel0 and channel1 are sampled on every positive edge of mclk. If..  
 -- either of these inputs switch high, P0\_valid\_data and P1\_valid\_data are set accordingly.

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if P0_DRDY = '1' then
            P0_valid_data <= '1';
        else
            P0_valid_data <= '0';
        end if;
    end if;
end process;

```

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if P1_DRDY = '1' then
            P1_valid_data <= '1';
        else
            P1_valid_data <= '0';
        end if;
    end if;
end process;

```

-- During acquisition mode P0\_Error and P1\_Error are sampled on every rising edge of clock edge.  
 -- If either of these lines switch high, an error has occurred. Such an error could be that a..  
 -- violation symbol was received by the FIFO/demuxes, or that data was incorrectly overwritten..  
 -- in the FIFO/demuxes. The transfer of data onto the RAU data bus is then disabled by driving..  
 -- Enable low.

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if op_mode(0) = '1' then
            if P0_Error = '1' or P1_Error = '1' then
                Enable <= '0';
            end if;
        else
            Enable <= '1';
        end if;
    end if;
end process;

```

-- When both sampling channels are active, data is transferred onto the RAU data bus from FIFO/..  
 -- demux0 and FIFO/demux1 alternatively. The signals that are generated to accomplish this in..  
 -- clude "toggle" and "handle". "Toggle" is toggled on each rising edge of mclk, producing a..  
 -- 25MHz square wave signal. In the 20 ns that toggle is high, either P0\_OE or P1\_OE can be..  
 -- activated, depending on "handle". If handle is low, P0\_OE is enabled. P1\_OE is enabled other..  
 -- wise. If an error is detected at any stage (Enable = '0') both P0\_OE and P1\_OE are inhibited.

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        toggle <= not(toggle);
    end if;
end process;

```

```

process(mclk)
begin
    if mclk'event and mclk = '1' then

```

```

if op_mode(0) = '1' then
    if Enable = '1' then
        if two_channels_active = '1' then
            if P0_valid_data = '1' and handle = '0' and toggle = '0' then
                handle <= '1';
            elsif P1_valid_data = '1' and handle = '1' and toggle = '0' then
                handle <= '0';
            end if;
        elsif channel1_active = '1' then
            handle <= '0';
        elsif channel0_active = '1' then
            handle <= '1';
        end if;
    else
        handle <= '0';
    end if;
else
    handle <= '0';
end if;
end process;

```

-- Only one of the output-enable lines to FIFO/demux0 and FIFO/demux1 can be enabled at a time.  
-- When there is valid data in FIFO/demux0 and channel 0 has the handle (handle = '0'), then..  
-- P0\_OE is activated in the time that toggle is high (for 20 ns). Similarly P1\_OE is activated..  
-- when handle = '1' and there is valid data in FIFO/demux1.

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if op_mode(0) = '0' then
            OE0 <= '1';
            OE1 <= '1';
        else
            if Enable = '1' then
                if P0_valid_data = '1' and handle = '0' and toggle = '1' then
                    OE0 <= '0';
                    OE1 <= '1';
                elsif P1_valid_data = '1' and handle = '1' and toggle = '1' then
                    OE0 <= '1';
                    OE1 <= '0';
                else
                    OE0 <= '1';
                    OE1 <= '1';
                end if;
            else
                OE0 <= '1';
                OE1 <= '1';
            end if;
        end if;
    end if;
end process;

```

```

P0_OE <= OE0;
P1_OE <= OE1;

```

```

-- *****
-- CONTROL OF RAM INTERFACE UNITS
-- *****

```

-- One\_ramblock\_active indicates whether only one or both of the DRAM blocks are active. If One..  
-- ramblock\_active is high, only one DRAM-module is implemented. If it is low then both DRAM-..

```

-- modules are implemented.
process(op_specs)
begin
    if op_specs(5 downto 4) = "10" then          -- only ramblock0 is implemented
        one_ramblock_active <= '1';
    else                                         -- ramblock0 & ramblock1 are implemented
        one_ramblock_active <= '0';
    end if;
end process;

-- When the system is in acquisition mode, data is transmitted through the RIUs from the RAU data..
-- bus to the DRAM modules. The direction input (DIR) to the RIUs is then set high. During down..
-- load mode the data is transferred from the DRAM modules back onto the RAU data bus. DIR is then..
-- set low.
DIR <= '1' when op_mode(0) = '1' else '0';

-- Cycle_counter is assigned cycle_acq during acquisition mode, and cycle_dwnl during download mode.
cycle_counter <= cycle_acq when op_mode(0) = '1'      -- acquisition mode
                else cycle_dwnl when op_mode(1) = '1'  -- download mode
                else '0';                             -- idle mode

-- Latchclk is assigned latch_acq during acquisition mode, and latch_dwnl during download mode.
Latchclk <= latch_acq when op_mode(0) = '1'          -- acquisition mode
            else latch_dwnl when op_mode(1) = '1'     -- download mode
            else '1';                                 -- idle mode

-- CS follows CS_acq during download mode, and CS_dwnl during download mode.
CS <= CS_acq when op_mode(0) = '1'                  -- acquisition mode
     else CS_dwnl when op_mode(1) = '0'             -- download mode
     else '0';                                       -- idle mode

-- During download mode RIU_OE is assigned RIU_OE_dwnl, whereas this signal is inactive during..
-- acquisition mode and idle mode.
RIU_OE <= RIU_OE_dwnl when op_mode(1 downto 0) = "10" else '0';

-- ***** ACQUISITION MODE *****

-- During acquisition mode "latch_acq" is assigned to the "Latchclk" output. Latchclk, which is..
-- active high, is then asserted as soon as any of P0_OE or P1_OE are deasserted. This means that..
-- just before the data on the RAU data bus becomes invalid, it is latched into one of the RIUs.
latch_acq <= OE0 and OE1;

-- During acquisition mode "cycle_acq" is assigned to "cycle_counter". "Cycle_acq" is toggled each..
-- time a word is latched into the RIUs. This signal is toggled on the negative edges of "latch_..
-- acq", ensuring that it is valid on the positive edge of "latch_acq".
process(latch_acq)
begin
    if latch_acq'event and latch_acq = '0' then
        cycle_acq <= not(cycle_acq);              -- toggle cycle_acq
    end if;
end process;

-- If both DRAM modules are implemented during acquisition mode, two 16-bit words are latched..
-- into each of the RIUs before the next RIU is accessed. CS_counter is incremented each time..
-- a byte is latched into an RIU (after OE0 is pulsed low). When CS_counter is 2, it means that..
-- two words have been latched into ramblock0. CS is then pulsed high, causing the next word to..
-- be latched into ramblock1. When CS_counter reaches 0, two words are latched into ramblock1.
-- CS is then driven low again.
process(mclk)
begin
    if mclk'event and mclk = '1' then

```

```

        if op_mode(0) = '1' then
            if OE0 = '0' or OE1 = '0' then
                CS_counter <= CS_counter + 1;
            end if;
        else
            CS_counter <= 0;
        end if;
    end if;
end process;

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if op_mode(0) = '1' then
            if Enable = '1' then
                if one_ramblock_active = '1' then
                    CS_acq <= '0';
                else
                    if CS_counter = 2 then
                        CS_acq <= '1';
                    elsif CS_counter = 0 then
                        CS_acq <= '0';
                    end if;
                end if;
            else
                CS_acq <= '0';
            end if;
        else
            CS_acq <= '0';
        end if;
    end if;
end process;

-- ***** DOWNLOAD MODE *****

-- During download mode, the data stored in the DRAM modules are read and transferred back to the..
-- RAU data bus. This process is controlled by means of a counter, namely dwnl_timer. This counter..
-- is incremented on each edge of mclk to 99, and then begins at zero again. Accordingly it cycles..
-- from 0 to 99 every 2 us.
process(mreset, mclk)
begin
    if mreset = '1' then
        dwnl_timer <= 0;
    elsif mclk'event and mclk = '1' then
        if op_mode(1 downto 0) = "10" then
            if dwnl_timer = 99 then
                dwnl_timer <= 0;
            else
                dwnl_timer <= dwnl_timer + 1;
            end if;
        else
            dwnl_timer <= 0;
        end if;
    end if;
end process;

-- In the first 1 us of a download cycle a 32-bit word is latched into the relevant RIUs from the..
-- implemented DRAM modules. The 16 most significant of these bits can be transferred to the RAU..
-- data bus when asserting RIU_OE. In the second microsecond of a download cycle, the 16 least..
-- significant bits of the 32-bit DRAM word can be accessed by asserting RIU_OE. A positive edge..
-- on latch_dwnl when cycle_dwnl is low causes the 32-bit DRAM word to be latched into the relevant..

```

```

-- RIU and the 16 MSBs to be made available to the RAU data bus. If cycle_dwnl is high on the..
-- positive edge of latch_dwnl, no new word is latched into the RIU, but the LSBs of the 32-bit..
-- DRAM word is made available to the RAU data bus.
process(mclk)
begin
    if mclk'event and mclk = '1' then
        case dwnl_timer is
            when 0 =>
                latch_dwnl <= '1';
                -- start of a download cycle
                -- access 16 MSBs of 32-bit DRAM word
            when 25 =>
                latch_dwnl <= '0';
                -- 0.5 us after start of download cycle
                -- clear Latchclk
            when 50 =>
                latch_dwnl <= '1';
                -- 1 us after start of download cycle
                -- access 16 LSBs of 32-bit word from DRAM
            when 75 =>
                latch_dwnl <= '0';
                -- 1.5 us after start of download cycle
                -- clear Latchclk
            when others =>
                latch_dwnl <= latch_dwnl;
        end case;
    end if;
end process;

-- Process that controls cycle_dwnl
process(mclk)
begin
    if mclk'event and mclk = '1' then
        case dwnl_timer is
            when 48 =>
                cycle_dwnl <= '1';
                -- Just before latch_dwnl is pulsed high
                -- 16 LSBs are to be made available to RAU data bus
            when 98 =>
                cycle_dwnl <= '0';
                -- Just before latch_dwnl is pulsed high
                -- New 32-bit word to be latched into RIUs
            when others =>
                cycle_dwnl <= cycle_dwnl;
        end case;
    end if;
end process;

-- In the first 0.5 microseconds of a download cycle, a 32-bit word is latched into RIU0 (and RIU1..
-- if both RAM blocks are active). The 16 most significant bits of the word in RIU0 is written onto..
-- the RAU data bus. In the second 0.5 microseconds, the 16 most significant bits of RIU1 transferred..
-- to the RAU data bus. If only one RAM block is implemented, the RIU0 is idle during this time.
-- During the third and fourth 0.5 microseconds of a download cycle, the 16 LSBs from RIU0 and RIU1..
-- are respectively transferred to the RAU data bus.
process(mclk, mreset)
begin
    if mreset = '1' then
        RIU_OE_dwnl <= '0';
    elsif mclk'event and mclk = '1' then
        if one_ramblock_active = '1' then
            case dwnl_timer is
                when 3 =>
                    RIU_OE_dwnl <= '1';
                when 53 =>
                    RIU_OE_dwnl <= '1';
                    -- RIU_OE pulsed high in first and third 0.5..
                    -- microseconds of download cycle
                when 15 =>
                    RIU_OE_dwnl <= '0';
                when 65 =>
                    RIU_OE_dwnl <= '0';
                    -- RIU_OE pulsed low 12 mclk-cycles after..
                    -- switching high.
                when others =>
                    end case;
            end case;
        else
            case dwnl_timer is

```

```

when 3 =>
    RIU_OE_dwnl <= '1';
when 28 =>
    RIU_OE_dwnl <= '1';
when 53 =>
    RIU_OE_dwnl <= '1';
when 78 =>
    RIU_OE_dwnl <= '1';
when 15 =>
    RIU_OE_dwnl <= '0';
when 40 =>
    RIU_OE_dwnl <= '0';
when 65 =>
    RIU_OE_dwnl <= '0';
when 90 =>
    RIU_OE_dwnl <= '0';
when others =>
end case;
end if;
end if;
end process;

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if one_ramblock_active = '1' then
            CS_dwnl <= '0';
        else
            case dwnl_timer is
                when 16 =>
                    CS_dwnl <= '1';
                when 66 =>
                    CS_dwnl <= '1';
                when 41 =>
                    CS_dwnl <= '0';
                when 91 =>
                    CS_dwnl <= '0';
                when others =>
            end case;
        end if;
    end if;
end process;

-- *****
-- CONTROL OF RAM CONTROLLERS
-- *****

-- During acquisition mode data is latched through the RIUs to the DRAM modules to be stored. After..
-- two 16-bit words have been latched through an RIU, the resulting 32-bit word is ready to be..
-- written into the relevant DRAM module. OE_counter toggles between 0 and 1 every time a byte is..
-- latched into an RIU. When OE_counter is 0 it means that two bytes have been latched into an RIU.
-- WR is then pulsed low for 20 ns. The 32-bit word on the relevant DRAM data bus is then written..
-- into memory.
process(mclk)
begin
    if mclk'event and mclk = '1' then
        if op_mode(0) = '1' then
            if OE0 = '0' or OE1 = '0' then
                OE_counter <= OE_counter + 1;
            if OE_counter = 0 then
                WR <= '0';
            end if;
        end if;
    end if;
end process;

```

-- RIU\_OE pulsed high in first, second, third..  
-- and fourth 0.5 microseconds of download cycle.

-- RIU\_OE pulsed low 12 mclk-cycles after..  
-- switching high.

-- simply RIU0 selected

-- RIU0 selected in first and third 0.5..  
-- microsecond intervals.

-- RIU1 selected in second and fourth..  
-- 0.5 microsecond intervals.

```

        else
            WR <= '1';           -- WR inactive
        end if;
    else
        WR <= '1';           -- WR inactive
    end if;
else
    WR <= '1';           -- WR inactive
end if;
end if;
end process;

```

-- During download mode the dwnl\_timer used for controlling the RIUs is also used for controlling..  
-- the RD signal to the DRAM controllers. The RD signal is pulsed low for one mclk-cycle, causing..  
-- a 32-bit word to be read from the implemented DRAM modules. This word is latched into the RIUs..  
-- and transferred to the RAU data bus by the RIUs.

```

process(mclk, mreset)
begin
    if mreset = '1' then
        RD <= '1';           -- RD inactive
    elsif mclk'event and mclk = '1' then
        if op_mode(1 downto 0) = "10" then           -- download mode
            if dwnl_timer = 4 then
                RD <= '0';           -- assert RD
            elsif dwnl_timer = 5 then
                RD <= '1';           -- RD inactive
            end if;
        else
            RD <= '1';           -- RD inactive
        end if;
    end if;
end process;

```

```

-- *****
--                               CONTROL OF RAM ADDRESS GENERATORS
-- *****

```

-- During acquisition mode the DRAM addresses are incremented after a 32-bit word has been..  
-- written into one of the DRAM modules.

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if op_mode(0) = '1' then           -- acquisition mode
            if OE0 = '0' or OE1 = '0' then           -- data transferred to RAU data bus
                if OE_counter = 0 then           -- two bytes latched through relevant RIU
                    INC_acq <= '1';           -- assert INC
                else
                    INC_acq <= '0';           -- INC deactivated
                end if;
            end if;
        end if;
    end if;
end process;

```

-- The dwnl\_timer is used for controlling the INC input to the DRAM address generators during..  
-- download mode. The DRAM address is incremented one mclk-cycle before a new 32-bit word is..  
-- read from the DRAM modules.

```

process(mclk)
begin
    if mclk'event and mclk = '1' then
        if op_mode(1 downto 0) = "10" then           -- download mode

```

```
        if dwnl_timer = 3 then
            INC_dwnl <= '1';           -- assert INC
        elsif dwnl_timer = 4 then
            INC_dwnl <= '0';           -- INC deactivated
        end if;
    else
        INC_dwnl <= '0';               -- reset INC to inactive level
    end if;
end if;
end process;

INC <= INC_acq when op_mode(0) = '1'  -- acquisition mode
    else INC_dwnl when op_mode(1) = '1' -- download mode
    else '0';                           -- inactive state

-- *****
--                                     EXTRAS
-- *****

BISTEN0 <= '1';
BISTEN1 <= '1';
ECP_PACK <= '0';
ECP_CLK <= '0';
TRIG <= '0';
reset_RAM <= '1' when op_mode(1 downto 0) = "00" else '0';

end A;
```

## **Appendix E**

### **Links to sites on programming Altera EPLDs and FPGAs**

#### **On-line literature: MAX7000 devices**

<http://www.altera.com/html/literature/lm7k.html>

Altera Programming Hardware Data Sheet, vers. 4

ByteBlaster Parallel Port Download Cable Data Sheet, ver. 1.01, June 1999

BitBlaster Serial Download Cable Data Sheet, ver. 4.02, June 1999

#### **On-line literature: FLEX 10K devices**

<http://www.altera.com/html/literature/lf10k.html#f10kds>

Configuring APEX 20K, FLEX 10K & FLEX 6000 Devices, ver. 1.01, August 1999

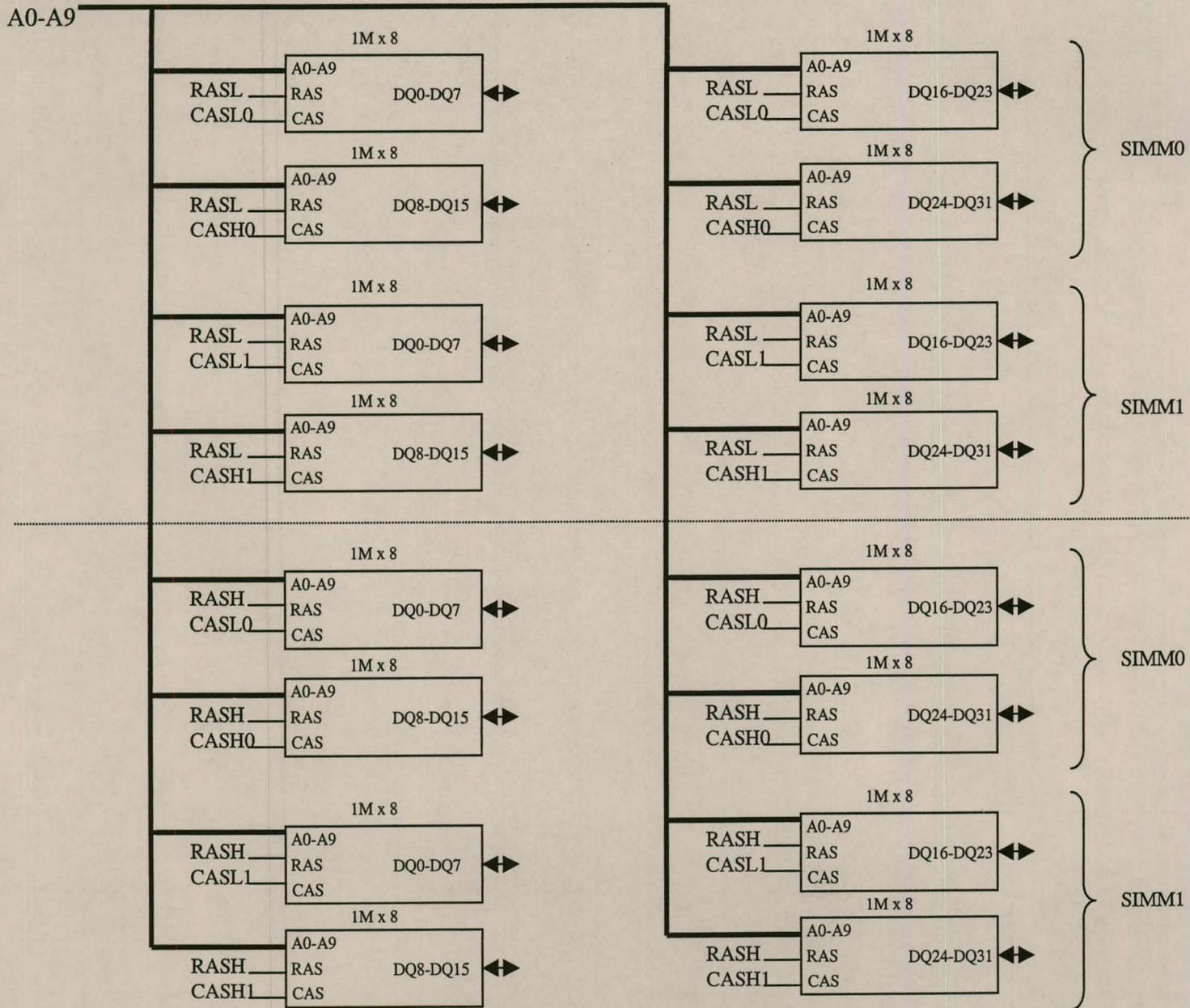
#### **On-line literature: FLEX 8000 devices**

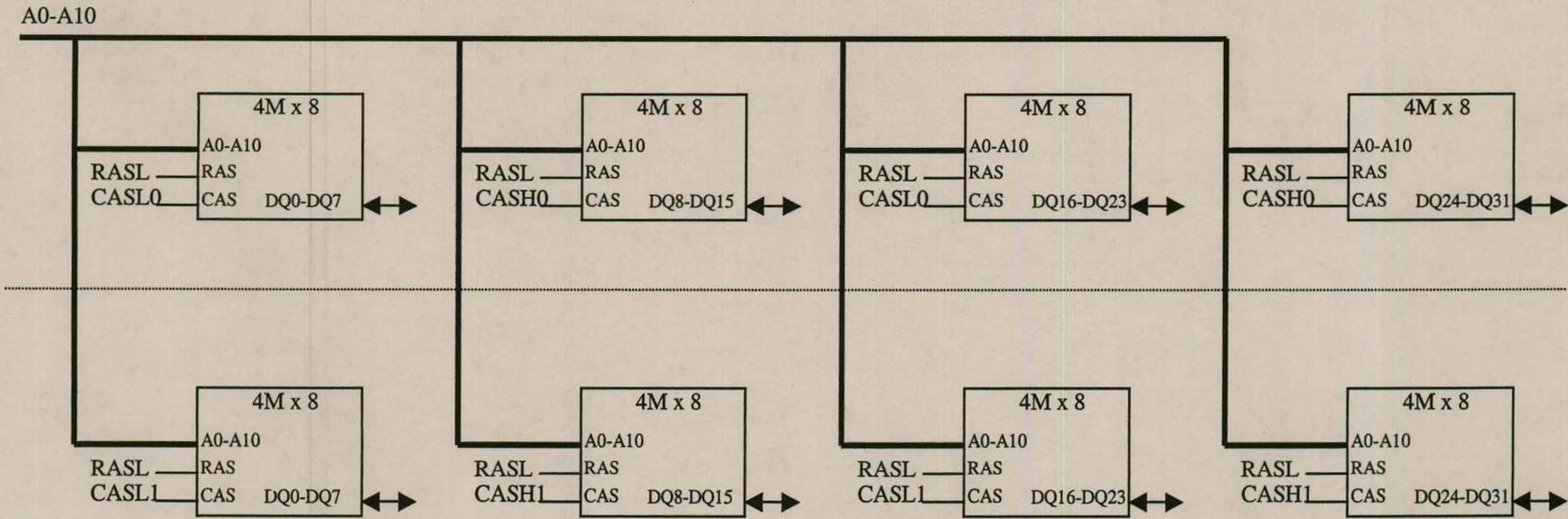
<http://www.altera.com/html/literature/lf8k.html>

Configuring FLEX 8000 Devices, ver. 3.01

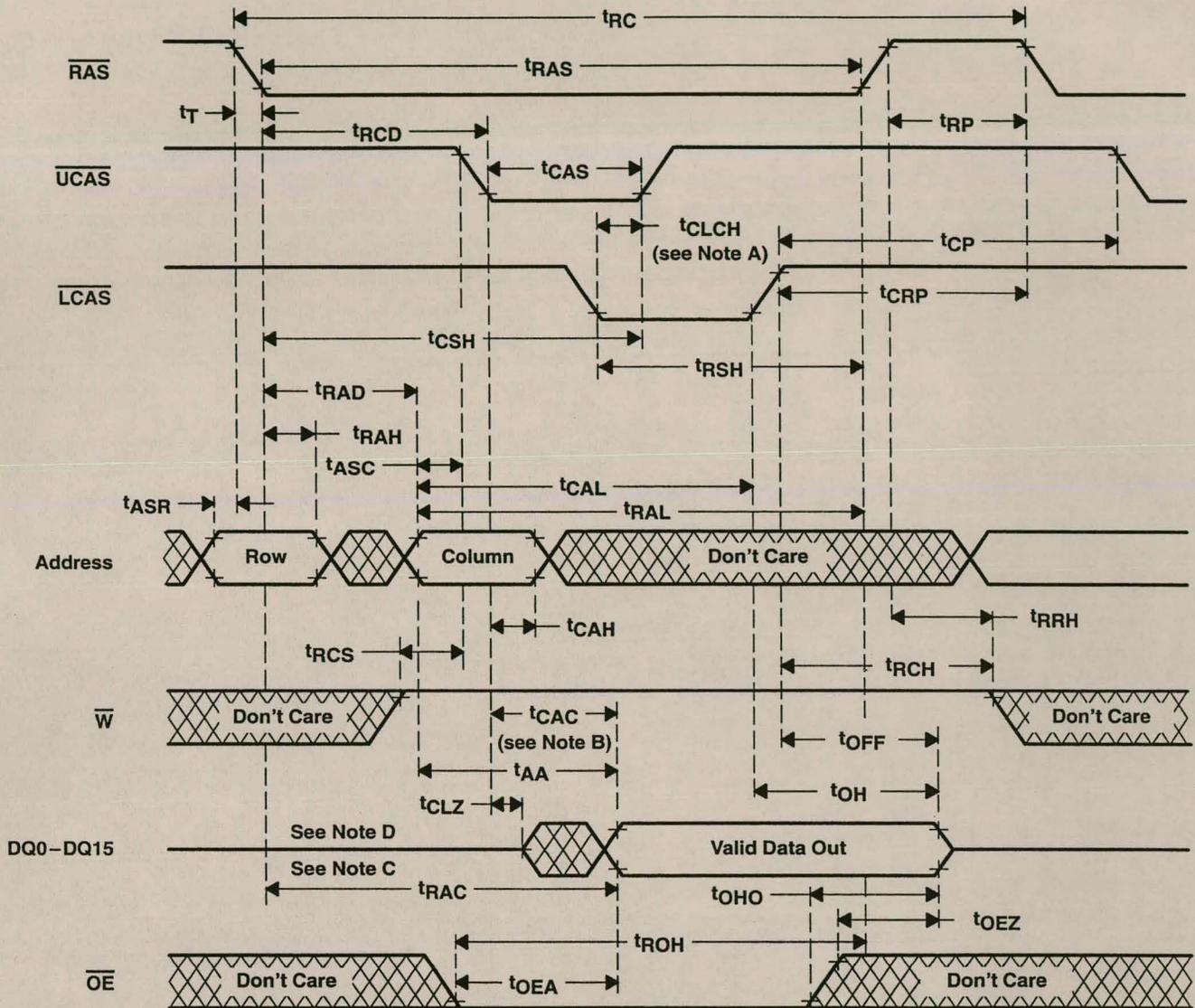
**Appendix F**  
**Details on 4, 8 and 16 Mbyte SIMMs in system**

**F1** Block diagram of 4/8 Mbyte SIMMs





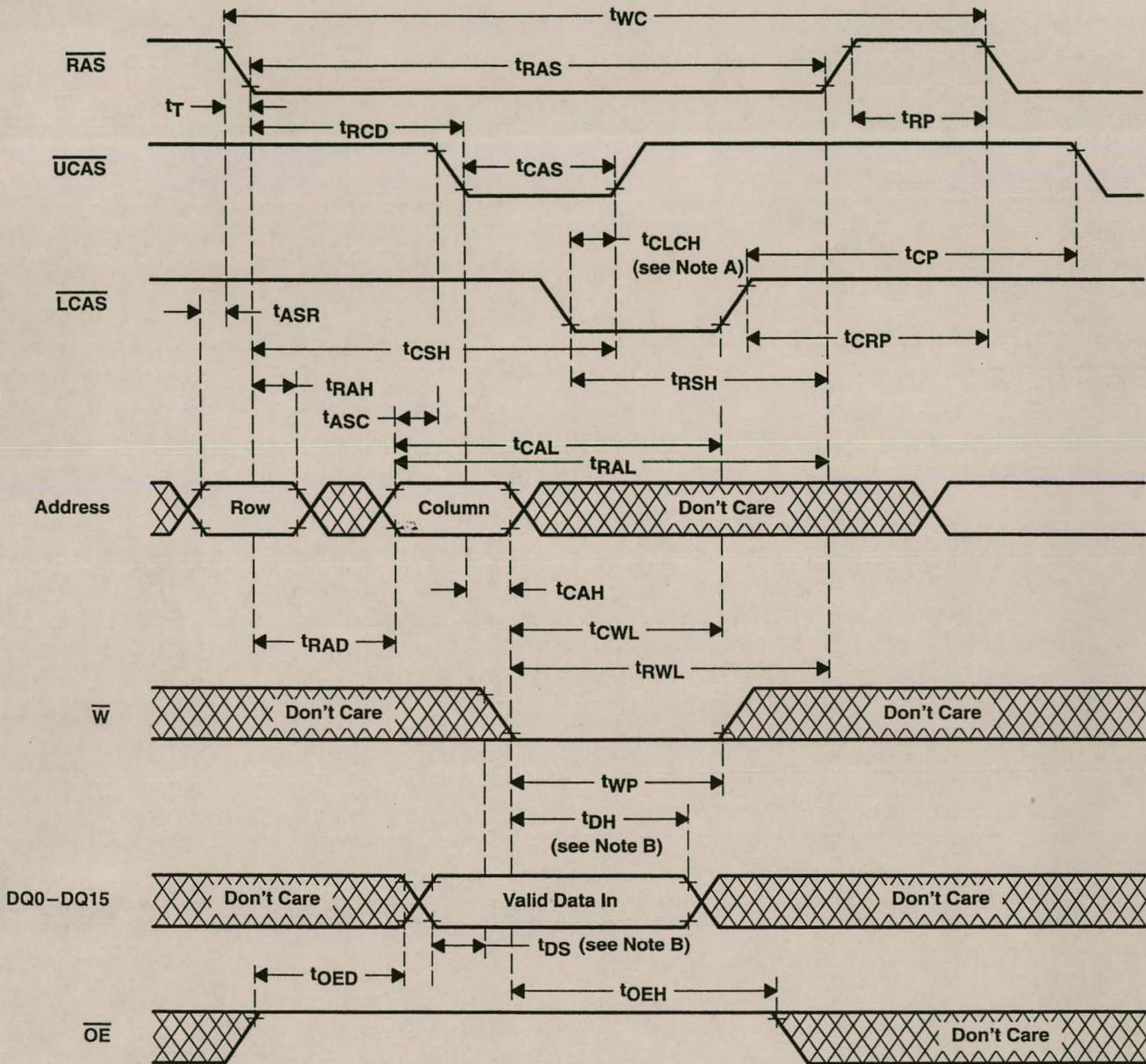
### F3 Timing waveforms for a read cycle



- NOTES: A. To hold the address latched by the first  $\overline{x}CAS$  going low, the parameter  $t_{CLCH}$  must be met.  
 B.  $t_{CAC}$  is measured from  $\overline{x}CAS$  to its corresponding DQx.  
 C. Output can go from the high-impedance state to an invalid-data state prior to the specified access time.  
 D.  $\overline{x}CAS$  order is arbitrary.

Figure 2. Read-Cycle Timing

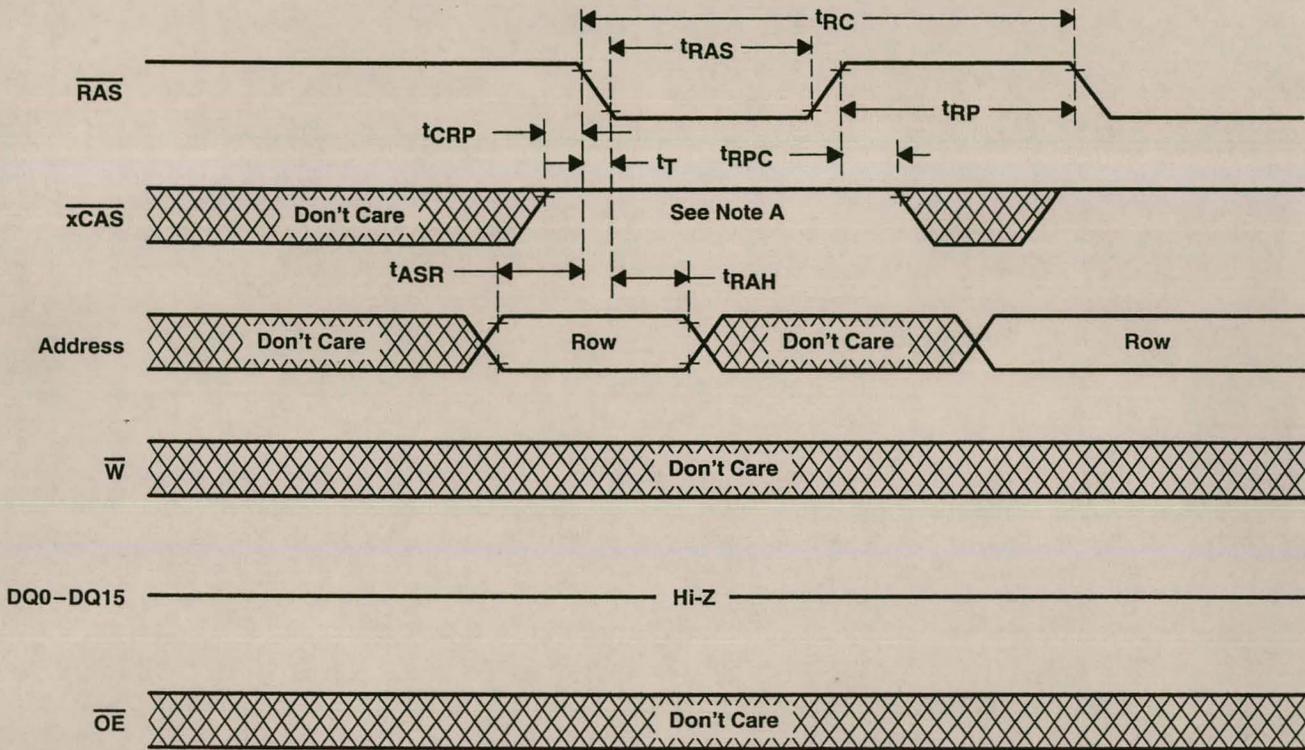
### F4 Timing waveforms for a write cycle



- NOTES: A. To hold the address latched by the first  $\overline{x}CAS$  going low, the parameter  $t_{CLCH}$  must be met.  
 B. Referenced to the first  $\overline{x}CAS$  or  $\overline{W}$ , whichever occurs last  
 C.  $\overline{x}CAS$  order is arbitrary.

Figure 3. Write-Cycle Timing

**F5 Timing waveforms for a refresh cycle**



NOTE A: All  $\overline{xCAS}$  must be high.

**Figure 9.  $\overline{RAS}$ -Only Refresh-Cycle Timing**

**F6 Read/write/refresh timing requirements**

Timing Parameter	Description	Specified range		Unit
		Min	Max	
t <sub>RC</sub>	Cycle time, random read or write	130		ns
t <sub>PC</sub>	Cycle time, page-mode read or write	45		ns
t <sub>RASP</sub>	Pulse duration, page-mode, RAS' low	70	100 000	ns
t <sub>RAS</sub>	Pulse duration, non-page-mode, RAS' low	70	10 000	ns
t <sub>CAS</sub>	Pulse duration, CAS' low	18	10 000	ns
t <sub>CP</sub>	Pulse duration, CAS' high	10		ns
t <sub>RP</sub>	Pulse duration, RAS' high (precharge)	50		ns
t <sub>WP</sub>	Pulse duration, WE' low	10		ns
t <sub>ASC</sub>	Setup time, column address before CAS' low	0		ns
t <sub>ASR</sub>	Setup time, row address before RAS' low	0		ns
t <sub>DS</sub>	Setup time, data before CAS' low	0		ns
t <sub>RCS</sub>	Setup time, WE' high before CAS' low	0		ns
t <sub>CWL</sub>	Setup time, WE' low before CAS' high	18		ns
t <sub>RWL</sub>	Setup time, WE' low before RAS' high	18		ns
t <sub>WCS</sub>	Setup time, WE' low before CAS' low	0		ns
t <sub>WRP</sub>	Setup time, WE' high before RAS' low	10		ns
t <sub>CAH</sub>	Hold time, column address after CAS' low	15		ns
t <sub>RHCP</sub>	Hold time, RAS' high from CAS' precharge	40		ns
t <sub>DH</sub>	Hold time, data after CAS' low	15		ns
t <sub>RAH</sub>	Hold time, row address after RAS' low	10		ns
t <sub>RCH</sub>	Hold time, WE' high after CAS' high	0		ns
t <sub>RRH</sub>	Hold time, WE' high after RAS' high	0		ns
t <sub>WCH</sub>	Hold time, WE' low after CAS' low	15		ns
t <sub>WRH</sub>	Hold time, WE' high after RAS' low	10		ns
t <sub>CHR</sub>	Delay time, RAS' low to CAS' high	10		ns
t <sub>CRP</sub>	Delay time, CAS' high to RAS' low	5		ns
t <sub>CSH</sub>	Delay time, RAS' low to CAS' high	70		ns
t <sub>CSR</sub>	Delay time, CAS' low to RAS' low	5		ns
t <sub>RAD</sub>	Delay time, RAS' low to column address	15	35	ns
t <sub>RAL</sub>	Delay time, column address to RAS' high	35		ns
t <sub>CAL</sub>	Delay time, column address to CAS' high	35		ns
t <sub>RCD</sub>	Delay time, RAS' low to CAS' low	20	52	ns
t <sub>RPC</sub>	Delay time, RAS' high to CAS' low	0		ns
t <sub>RSH</sub>	Delay time, CAS' low to RAS' high	18		ns
t <sub>REF</sub>	Refresh time interval		32	ns
t <sub>T</sub>	Transition time	3	30	ns

## Appendix G

### AT89C2051 Microcontroller Buffers & Program

#### G1 AT89C2051 internal buffers

<b>ACC</b>	Accumulator Register
<b>B</b>	B Register For multiply and divide operations
<b>SP</b>	Stack Pointer Starts at location 08H
<b>DPH</b>	Data Pointer High Holds high part of a 16-bit address
<b>DPL</b>	Data Pointer Low Holds low part of a 16-bit address
<b>P1</b>	Port1
<b>P3</b>	Port3
<b>SBUF</b>	Serial Port Buffer
<b>TL0 &amp; TH0</b>	16-bit Counter Register For Timer/Counter 0
<b>TL1 &amp; TH1</b>	16-bit Counter Register For Timer/Counter1
<b>PSW</b>	Program Status Word

<b>CY</b>	<b>AC</b>	<b>F0</b>	<b>RS1</b>	<b>RS0</b>	<b>OV</b>	<b>-</b>	<b>P</b>
-----------	-----------	-----------	------------	------------	-----------	----------	----------

CY – Carry Flag  
 AC – Auxiliary Carry Flag (For BDC operations)  
 F0 – Flag0 (For general purposes)  
 RS1 & RS0 – Choose Working Register Bank (p 2-23, 2-42)  
 OV – Overflow Flag  
 P – Parity Flag (Indicates even or odd number of 1's in ACC)

**TMOD**      Timer/Counter Mode Register

<b>GATE</b>	<b>C/T'</b>	<b>M1</b>	<b>M0</b>	<b>GATE</b>	<b>C/T'</b>	<b>M1</b>	<b>M0</b>
<b>Timer 1</b>				<b>Timer 0</b>			

Gate – 0 : Timer X is enable when TR X in TCON is 1.  
       1 : Timer X is enabled when INT X input pin 1 and TR X in TCON 1.

C/T' – Timer (input from internal system clock) or Counter (input from Tx input pin) Select.

M1 – Mode bit 1 (p. 2-46)

M0 – Mode bit 0 (p. 2-46)

## **TCON**      Timer/Counter Control Register

<b>TF1</b>	<b>TR1</b>	<b>TF0</b>	<b>TR0</b>	<b>IE1</b>	<b>IT1</b>	<b>IE0</b>	<b>IT0</b>
------------	------------	------------	------------	------------	------------	------------	------------

TF1 – Timer 1 Overflow flag

TR1 – Timer 1 Run control bit. Set/cleared to turn timer/counter on/off.

TF0 – Timer 0 Overflow flag.

TR0 – Timer 0 Run control bit.

IE1 – Interrupt 1 edge flag. Set when INT0 edge detected. Cleared when interrupt serviced.

IT1 – Interrupt 1 type control. Set/cleared to specify falling edge/low level triggered interrupt.

IE0 – Interrupt 0 edge flag.

IT0 – Interrupt 0 type control.

## **SCON**      Serial Port Control Register

<b>SM0</b>	<b>SM1</b>	<b>SM2</b>	<b>REN</b>	<b>TB8</b>	<b>RB8</b>	<b>T1</b>	<b>RI</b>
------------	------------	------------	------------	------------	------------	-----------	-----------

SM0 – Serial Port Mode bit 0 (p. 2-51)

SM1 – Serial Port Mode bit 1 (p. 2-51)

SM2 – For Multiprocessor Communication (p. 2-51)

REN – Enables serial reception

TB8 – 9<sup>th</sup> Data bit that will be transmitted in Mode 2 and 3

RB8 – 9<sup>th</sup> Data bit that was received in Mode 2 and 3

T1 – Transmit interrupt flag. Set by hardware. Cleared by software (p.2-51)

RI – Receive interrupt flag. Set by hardware. Cleared by software (p. 2-51)

## **IE**      Interrupt Enable Register

<b>EA</b>	<b>-</b>	<b>-</b>	<b>ES</b>	<b>ET1</b>	<b>EX1</b>	<b>ET0</b>	<b>EX0</b>
-----------	----------	----------	-----------	------------	------------	------------	------------

EA – Enables/disables all interrupts

ES – Serial Port interrupt enable bit

ET1 – Timer 1 interrupt enable bit

EX1 – External interrupt 1 enable bit

ET0 – Timer 0 interrupt enable bit

EX0 – External interrupt 0 enable bit

## **IP**      Interrupt Priority Register

<b>-</b>	<b>-</b>	<b>-</b>	<b>PS</b>	<b>PT1</b>	<b>PX1</b>	<b>PT0</b>	<b>PX0</b>
----------	----------	----------	-----------	------------	------------	------------	------------

PS – Serial Port interrupt priority bit

PT1 – Timer 1 interrupt priority bit

PX1 – External interrupt 1 priority bit

PT0 – Timer 0 interrupt priority bit

PX0 – External interrupt 0 priority bit



```

{Timer0 16-bit mode}

{TCON Register(not TF1,TF0,}
{TR0, IE1, IT1, IE0, IT0}
TR1 := true;                                {Switch on Timer 1}

{SCON Register}
SCON := $50;                                {Serial Port 8-bit UART}

{IE (not EX1, ET0, EX0)}
ES := true;                                {Enable serial port interrupts}
ET1 := false;                              {Disable Timer 1 interrupts}
{EX1 := false;}
{EX0 := false;}
{ET0 := false;}

{IP (not PS, PX1, PT0, PX0)}
PT1 := true;                                {Timer 1 highest interrupt priority}

{PCON}
PCON := $80;                                {Double Baud Rate}
EA := true;                                 {Enable all interrupts}
end;

procedure interrupt Serial; using 2;
var
  i, j : integer;
begin
  if RI then begin
    RI := false;
    if (PCON and $08) = $00 then begin        {PCON.3 = 0}
      PCON := (PCON or $08);                {PCON.3 := 1}
      if SBUF = $80 then begin              {MSB of SBUF is high}
        PCON := (PCON or $04);             {PCON.2 := 1}
      end
    else begin
      PCON := (PCON and $FB);              {PCON.2 := 0}
    end;
  end
  else begin
    PCON := (PCON and $F7);                {PCON.3 := 0}
    upbus := SBUF;
    if (PCON and $04) = $00 then begin     {PCON.2 = 0}
      {for i := 1 to 100 do;}
      {for j := 1 to 50 do;}
      WR := true;
      {for i := 1 to 100 do;}
      {for j := 1 to 50 do;}
      WR := false;
    end
  end
end

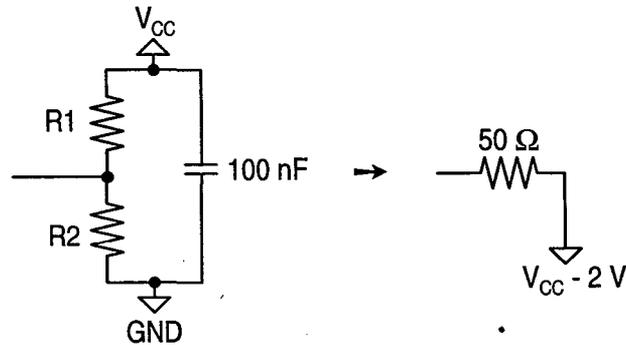
```

```
else begin
    {for i := 1 to 100 do;}
    {for j := 1 to 50 do;}                {PCON.2 = 1}
    ALE := true;
    {for i := 1 to 100 do;}
    {for j := 1 to 50 do;}
    ALE := false;
end;
end;
end;
end;

begin
    initialise;
    repeat until false;
end.
```

## Appendix H

### Deduction of equations (3.1) and (3.2)



The Thevenin equivalent of the first circuit in the figure above is represented as follows:

$$R_{TH} = \frac{R_1 R_2}{R_1 + R_2} = 50\Omega \quad (\text{H.1})$$

$$V_{TH} = \frac{R_2}{R_1 + R_2} V_{CC} = V_{CC} - 2 \quad (\text{H.2})$$

From equation (H.2):

$$\begin{aligned} (V_{CC} - 2)(R_1 + R_2) &= R_2 V_{CC} \\ R_1 V_{CC} + R_2 V_{CC} - 2R_1 - 2R_2 &= R_2 V_{CC} \\ R_2 &= \frac{R_1}{2} (V_{CC} - 2) \end{aligned} \quad (\text{H.3})$$

Substitute (H.3) into (H.1):

$$\begin{aligned} 50 &= \frac{R_1 \frac{R_1}{2} (V_{CC} - 2)}{R_1 + \frac{R_1}{2} (V_{CC} - 2)} \\ 50 &= \frac{R_1 (V_{CC} - 2)}{2 + (V_{CC} - 2)} \\ R_1 &= \frac{50 V_{CC}}{V_{CC} - 2} \end{aligned} \quad (\text{H.4})$$

Substitute (H.4) into (H.3):

$$\begin{aligned} R_2 &= \frac{\frac{50 V_{CC}}{V_{CC} - 2}}{2} (V_{CC} - 2) \\ R_2 &= \frac{50 V_{CC}}{2} \end{aligned} \quad (\text{H.5})$$

# Appendix I

## Software control program code & MATLAB program

### I1 Software control program

```
unit DataAcq;

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    StdCtrls, ComDrv32;

type
    IntPtr = ^Integer;
    TDataAcq = class(TForm)
        Button2: TButton;
        CheckBox1: TCheckBox;
        CheckBox2: TCheckBox;
        Label1: TLabel;
        Label2: TLabel;
        CheckBox3: TCheckBox;
        CheckBox4: TCheckBox;
        Label3: TLabel;
        ComboBox1: TComboBox;
        Button3: TButton;
        CommPortDriver1: TCommPortDriver;
        Button4: TButton;
        Label5: TLabel;
        Button1: TButton;

    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);

    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    DataAcq: TDataAcq;
    BytePtr : IntPtr;

implementation
```

```
{$R *.DFM}
```

```
uses
```

```
WarningForm;
```

```
procedure ByteSend(A : byte);
```

```
begin
```

```
new(BytePtr);
```

```
BytePtr^ := A;
```

```
DataAcq.CommPortDriver1.SendData(BytePtr, 1);
```

```
dispose(BytePtr);
```

```
end;
```

```
{EXIT BUTTON}
```

```
procedure TDataAcq.Button1Click(Sender: TObject);
```

```
begin
```

```
CommPortDriver1.Connect;
```

```
{SET OPERATING MODE OF SYSTEM TO IDLE MODE}
```

```
ByteSend(128); {ALE must be pulsed}
```

```
ByteSend(10); {00001010 - Select op_mode register}
```

```
ByteSend(0); {WR must be pulsed}
```

```
ByteSend(0); {Switch to IDLE MODE}
```

```
{RESET ADDRESS GENERATOR 0}
```

```
ByteSend(128); {ALE must be pulsed}
```

```
ByteSend(160); {10100000 - Select address generator 0}
```

```
ByteSend(0); {WR must be pulsed}
```

```
ByteSend(1); {00000001 - Reset address generator 0}
```

```
{RESET ADDRESS GENERATOR 1}
```

```
ByteSend(128); {ALE must be pulsed}
```

```
ByteSend(175); {10101111 - Select address generator 1}
```

```
ByteSend(0); {WR must be pulsed}
```

```
ByteSend(1); {00000001 - Reset address generator 1}
```

```
CommPortDriver1.Disconnect;
```

```
DataAcq.Visible := false;
```

```
Form1.Visible := true;
```

```
end;
```

```
{IDLE MODE SELECTION BUTTON}
```

```
procedure TDataAcq.Button3Click(Sender: TObject);
```

```
begin
```

```
CommPortDriver1.Connect;
```

```
{SET OPERATING MODE OF SYSTEM TO IDLE MODE}
```

```
ByteSend(128); {ALE must be pulsed}
```

```
ByteSend(10); {00001010 - Select op_mode register}
```

```
ByteSend(0); {WR must be pulsed}
```

```
ByteSend(0); {Switch to IDLE MODE}
```

```
{RESET ADDRESS GENERATOR 0}
```

```
ByteSend(128); {ALE must be pulsed}
```

```
ByteSend(160); {10100000 - Select address generator 0}
```

```
ByteSend(0); {WR must be pulsed}
```

```

ByteSend(1);          {00000001 - Reset address generator 0}
{RESET ADDRESS GENERATOR 1}
ByteSend(128);       {ALE must be pulsed}
ByteSend(175);       {10101111 - Select address generator 1}
ByteSend(0);         {WR must be pulsed}
ByteSend(1);         {00000001 - Reset address generator 1}
CommPortDriver1.Disconnect;

```

end;

{DOWNLOAD BUTTON}

```

procedure TDataAcq.Button4Click(Sender: TObject);

```

```

begin

```

```

CommPortDriver1.Connect;
{SET OPERATING MODE OF SYSTEM TO IDLE MODE}
ByteSend(128);       {ALE must be pulsed}
ByteSend(10);        {00001010 - Select op_mode register}
ByteSend(0);         {WR must be pulsed}
ByteSend(0);         {Switch to IDLE MODE}
{RESET ADDRESS GENERATOR 0}
ByteSend(128);       {ALE must be pulsed}
ByteSend(160);       {10100000 - Select address generator 0}
ByteSend(0);         {WR must be pulsed}
ByteSend(1);         {00000001 - Reset address generator 0}
{RESET ADDRESS GENERATOR 1}
ByteSend(128);       {ALE must be pulsed}
ByteSend(175);       {10101111 - Select address generator 1}
ByteSend(0);         {WR must be pulsed}
ByteSend(1);         {00000001 - Reset address generator 1}
{ENABLE COUNTING AND TRIGGERING BY ADDRESS GENERATOR 0}
ByteSend(128);       {ALE must be pulsed}
ByteSend(160);       {10100000 - Select address generator 0}
ByteSend(0);         {WR must be pulsed}
ByteSend(2);         {00000010 - Enable counting and triggering}
{ENABLE COUNTING AND TRIGGERING BY ADDRESS GENERATOR 1}
ByteSend(128);       {ALE must be pulsed}
ByteSend(175);       {10101111 - Select address generator 1}
ByteSend(0);         {WR must be pulsed}
ByteSend(2);         {00000010 - Enable counting}
{SET CONTROL MODULE INTO DOWNLOAD MODE}
ByteSend(128);       {uC must pulse ALE}
ByteSend(10); - - -   {00001010 - Select op_mode register}
ByteSend(0);         {uC must pulse WR}
ByteSend(2);         {Set RAU control module into download mode}
CommPortDriver1.Disconnect;

```

end;

{ACQUISITION BUTTON - SET SAMPLING FREQUENCY AND NUMBER OF SAMPLES}

{SET RAU CONTROL MODULE INTO ACQUISITION MODE}

```

procedure TDataAcq.Button2Click(Sender: TObject);

```

```

var

```

```

channel0, channel1 : boolean;
ramblock0, ramblock1 : boolean;
op_specs : byte;
begin
  {DETERMINE IF AT LEAST ONE PROBE AND ONE RAMBLOCK HAVE BEEN ACTIVATED}
  channel0 := CheckBox1.Checked;
  channel1 := CheckBox2.Checked;
  ramblock0 := CheckBox3.Checked;
  ramblock1 := CheckBox4.Checked;
  if not ((channel0 or channel1) and (ramblock0 or ramblock1)) then
    begin
      DataAcq.Enabled := false;
      WarningForm.Label1.Caption := 'Enter at least one Active Channel';
      WarningForm.Visible := true;
      exit;
    end
  {SEND ACQUISITION MODE COMMANDS}
  else
    begin
      case ComboBox1.ItemIndex of
        0: op_specs := 5;
        1: op_specs := 4;
        2: op_specs := 3;
        3: op_specs := 2;
        4: op_specs := 1;
        5: op_specs := 0;
      else
        op_specs := 5;
      end;
      if CheckBox1.Checked then
        op_specs := op_specs + 128;
      if CheckBox2.Checked then
        op_specs := op_specs + 64;
      if CheckBox3.Checked then
        op_specs := op_specs + 32;
      if CheckBox4.Checked then
        op_specs := op_specs + 16;
      CommPortDriver1.Connect;
      {SET OPERATING MODE OF SYSTEM TO IDLE MODE}
      ByteSend(128);      {ALE must be pulsed}
      ByteSend(10);      {00001010 - Select op_mode register}
      ByteSend(0);      {WR must be pulsed}
      ByteSend(0);      {Switch to IDLE MODE}
      {RESET ADDRESS GENERATOR 0}
      ByteSend(128);      {ALE must be pulsed}
      ByteSend(160);      {10100000 - Select address generator 0}
      ByteSend(0);      {WR must be pulsed}
      ByteSend(1);      {00000001 - Reset address generator 0}
      {RESET ADDRESS GENERATOR 1}
      ByteSend(128);      {ALE must be pulsed}
    end
  end;
end;

```

```

ByteSend(175);      {10101111 - Select address generator 1}
ByteSend(0);        {WR must be pulsed}
ByteSend(1);        {00000001 - Reset address generator 1}
{ENABLE COUNTING BY ADDRESS GENERATOR 0}
ByteSend(128);      {ALE must be pulsed}
ByteSend(160);      {10100000 - Select address generator 0}
ByteSend(0);        {WR must be pulsed}
ByteSend(2);        {00000010 - Enable incrementing of address}
{ENABLE COUNTING BY ADDRESS GENERATOR 1}
ByteSend(128);      {ALE must be pulsed}
ByteSend(175);      {10101111 - Select address generator 1}
ByteSend(0);        {WR must be pulsed}
ByteSend(2);        {00000010 - Enable incrementing of address}
{SET OPERATING SPECIFICATIONS AS SELECTED BY THE USER}
ByteSend(128);      {ALE must be pulsed}
ByteSend(15);       {00001111 - Select op_specs register}
ByteSend(0);        {WR must be pulsed}
ByteSend(op_specs); {Send operating specifications}
{SET OPERATING MODE TO ACQUISITION MODE}
ByteSend(128);      {uC must pulse ALE}
ByteSend(10);       {Select op_mode register}
ByteSend(0);        {uC must now pulse WR}
ByteSend(1);        {Set control module into acquisition mode}
CommPortDriver1.Disconnect;
end;
end;
end.

unit WarningForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TWarningForm = class(TForm)
    Button1: TButton;
    Label1: TLabel;

  procedure Button1Click(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
end;

```

```

var
    WarningForm: TWarning;
implementation

{$R *.DFM}

uses
    DataAcq;

procedure TWarning.Button1Click(Sender: TObject);
begin
    Warning.Visible := false;
    DataAcq.Visible := true;
    DataAcq.Enabled := true;
end;

end.

```

## I2 MATLAB program for viewing *Jobmatch* data

```

% log3rd.m
fid = fopen('RAUa10ks.txt','r');

for c1 = 1:5000 %number of points to plot
    for c2 = 1:8
        A = fread(fid,5);
        a((c1-1)*8+c2) = hex2dec(sym(setstr(A(1:4))));
    end;
    fread(fid,2);
end;

fclose(fid)

figure(1)
plot(a)

b=(a-2.070e4)/max(a-2.07e4);

for k = 1:length(b)
    if b(k)<0
        b(k)=0;
    end;
end;

figure(2)
plot(b)

```

```
title('DATA SAMPLED ON RAU DATA BUS DURING DOWNLOAD MODE');  
xlabel('Data points');  
ylabel('Normalised amplitude');
```