M68EZ328ADS v2.0
Application Development System

User's Manual
Revision 1.0

JAN 19, 2000

Preliminary

TABLE OF CONTENTS

Section 1
General Information

Section 2
Quick Installation Guide

Section 3
Hardware Description and Board Operation

Table of Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1
GENERAL INFORMATION

1.1 INTRODUCTION

The DragonBall™-EZ (MC68EZ328) Application Development System (M68EZ328ADS) is designed to supply users with an environment to develop MC68EZ328 based application software. Moreover, this board can be used as a reference for real-life product design. M68EZ328ADS provides several interface ports for application software and target board debug purpose.

DragonBall™-EZ ADS altogether has two versions released for customers. This is the second version. The board design and layout in second version has been changed a great deal. It includes simplifying the design for enhancing production quality and the easiness to use. However, most of the interfaces to external peripherals remain unchanged in order to provide high degree of compatibility with previous version. For more details on the changes, Please refer to Appendix A .

This document will discuss the usage and system details of the M68EZ328ADS v2.0.

1.2 FEATURES

- MC68EZ328 CPU
- Memory Subsystem

  —2 MB FLASH (Expandable to 4MB)
  —8 MB EDO DRAM
- Debug ports

  —One RS232 serial port interface to MC68EZ328 internal UART
  —One External RS232 serial port connecting to on-board UART chip
  —Direct logic analyzer interface to system bus
- LCD and Touch Panel Interface

  —MC68EZ328 LCD interface
  —Burr-Brown 12-bit touch panel controller ADS7843E available for pen input
- LED Indicators

  —Red LED for power
  —Green LED for system heart beat
  —Yellow LED for status of MC68EZ328 pin PD0
  —Yellow LED for status of MC68EZ328 pin PD1
- Board operation mode support

  —MC68EZ328 EMU mode
  —MC68EZ328 normal mode
  —MC68EZ328 bootstrap mode
- Debug Monitor

  —MetroWerks Codewarrior Target Monitor using serial port

—SDS source-level debugger monitor by Software Development System Inc. using serial port

- Clock Source

—32.768KHz for MC68EZ328 internal PLL

- Power Supply

—3.0V - 3.3V main power supply

1.3 RELATED DOCUMENTATION

The following documents can be used as references when using M68EZ328ADS.

- MC68EZ328 User's Manual
- MC68EZ328 Product Information

1.4 TECHNICAL SUPPORT

1.4.1 M68EZ328ADS v2.0

For getting the latest information, please visit our web page:

`http://www.mot.com/SPS/WIRELESS/products/DragonBall.html`

`http://www.apspg.com/products/dragonballez/mc68ez328.html`

1.4.2 Debugger

There are three source-level debuggers for DragonBall™-EZ. The contact information is listed below.

1. Metrowerks

   `http://www.metrowerks.com`

2. Single Step Development

   `http://www.sdsi.com`

3. Microtek SLD

   `http://www.microtekintl.com`

SECTION 2
QUICK INSTALLATION GUIDE

2.1 OVERVIEW

This section provides a description of the evaluation module, requirements, quick installation and test information. Detailed information on the M68EZ328ADS v2.0 design and operation is provided in the remaining sections of this manual.

2.2 EQUIPMENTS REQUIRED

The following equipments are required to use with the M68EZ328ADS Application Development System, some of them are already bundled with the ADS package.

- Power supply – 3.0V-3.3V, 1500mA, with 2 mm female (inside positive) power connector
- RS-232 cable (DB9 male to DB9 female)
- IBM PC compatible computer (486 class or higher) running Windows 3.1 and DOS 6.0 (or higher), or Windows 95, with an RS-232 serial port capable of 9600-115200 bit per second operation

2.3 INSTALLATION PROCEDURE

Please follow the procedure below to set up M68EZ328ADS.

1. Prepare the M68EZ328ADS v2.0 board
2. Connect the M68EZ328ADS v2.0 board to PC and power supply
3. Install software debugger

2.3.1 Prepare the M68EZ328ADS v2.0 board

Locate the DIP switches on the M68EZ328ADS v2.0 board and select appropriate monitor and debug port for your debugger.

Figure 2-2 shows the factory default DIP switches settings. This setting selects to use Metroworks monitor. Figure 2-3 shows the DIP switches setting of using SDS monitor.

For detail description of each switch. Pls. refer to Table 3-1 and Table 3-2.

For additional information on the M68EZ328ADS v2.0 and its components. Pls refer to Section 3 .

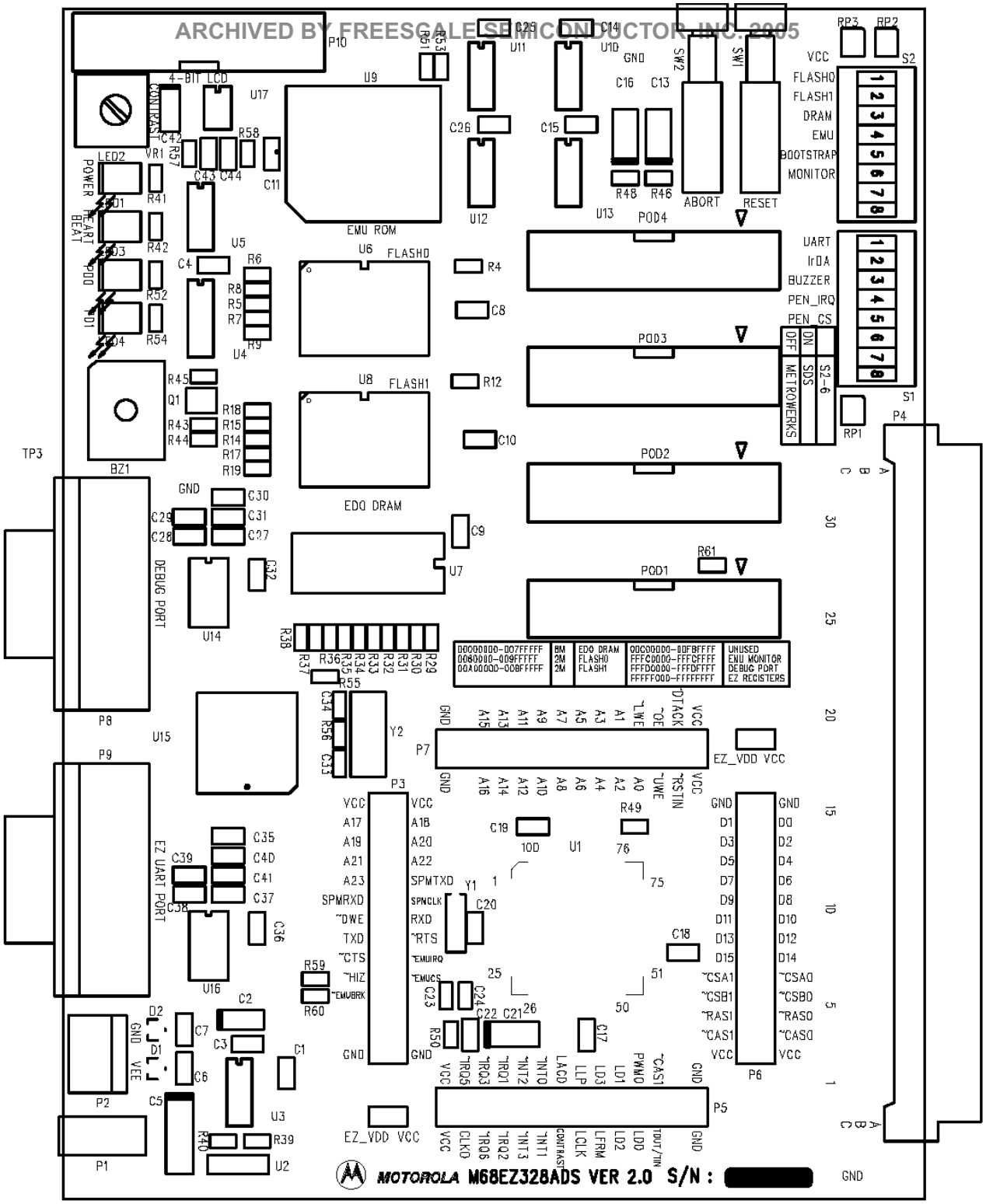


Figure 2-1. M68EZ328ADS v2.0 Key Component Layout

Monitor and Debug port selection

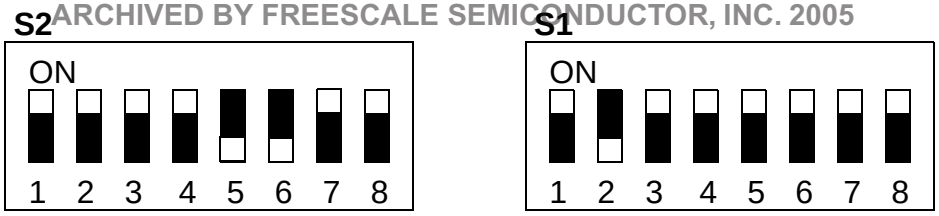


Figure 2-2. Default DIP Switch Options for Metrowerks monitor

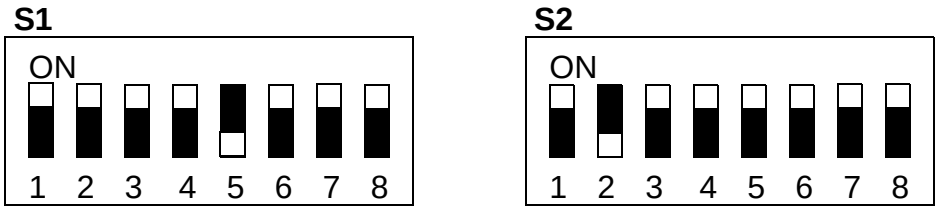


Figure 2-3. DIP Switch Options for SDS monitor

2.3.2 Connecting M68EZ328ADS v2.0 to PC

Figure 2-4 shows connections among the PC, the external power supply and the M68EZ328ADS v2.0 board. Use the following steps to complete cable connections:

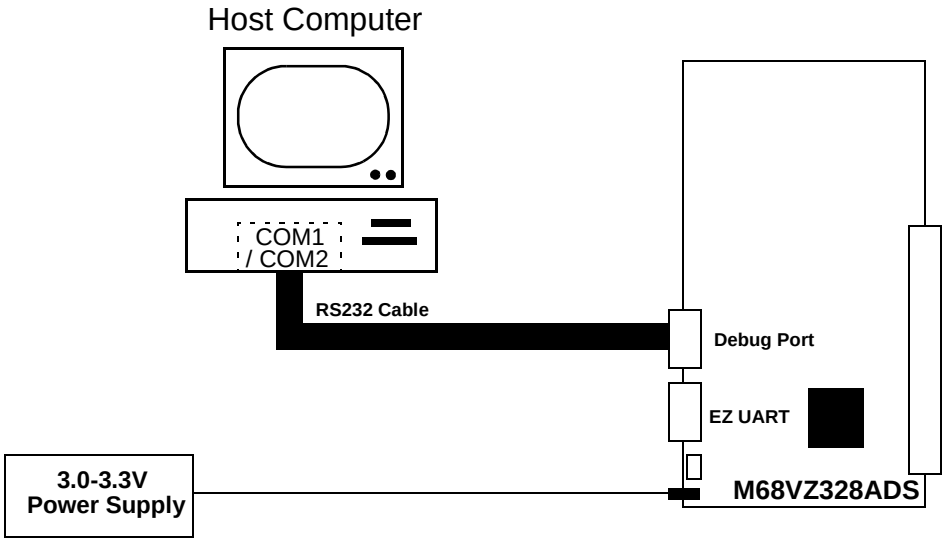


Figure 2-4. Connecting PC to Debug Port of M68EZ328ADS v2.0

For most evaluation platforms, serial communication is the primary channel to link up PC with the target board. Both Metrowerks Codewarrior and Software Development System (SDS) debug monitor support serial communication through UART port. The procedure are as follows.

1. Connect a RS232 cable from COM port (COM1 or COM2) to connector P2 of M68EZ328ADS v2.0.
2. Connect the power supply +3V or power adaptor to the P1 of M68EZ328ADS v2.0
3. Turn on the power supply. The RED LED will flash and the GREEN LED will illuminate when power is correctly applied.

2.3.3 Installing software debugger

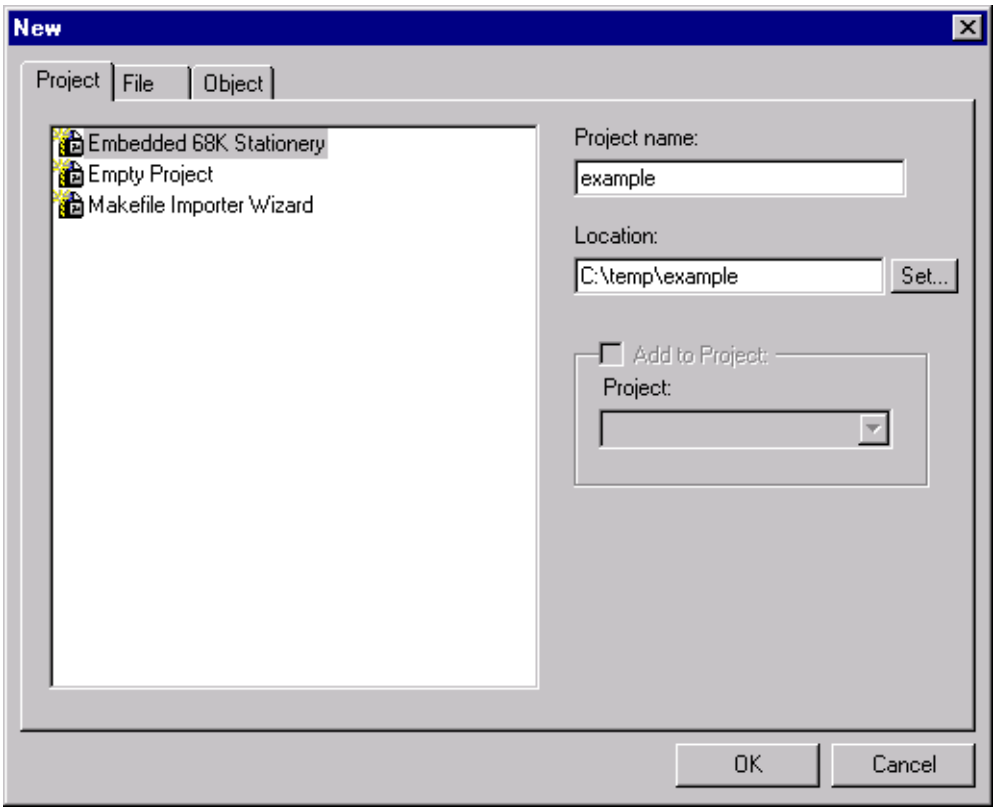
The following software debugger supports M68EZ328ADS v2.0:

- Metrowerks Codewarrior
- Single Step Development System
- SLD

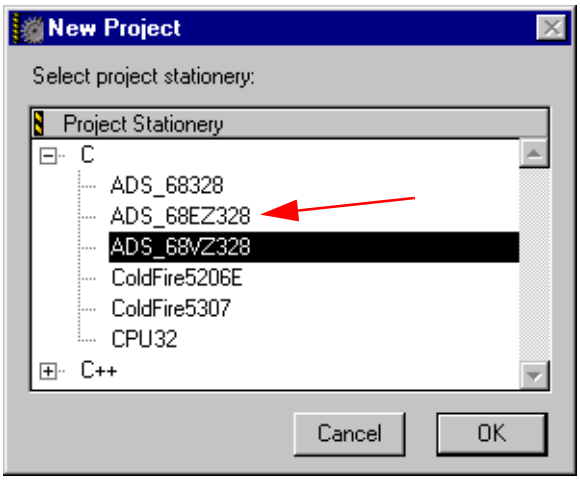
Metrowerks Codewarrior

A simple procedure for using Metrowerks Codewarrior Target Monitor:

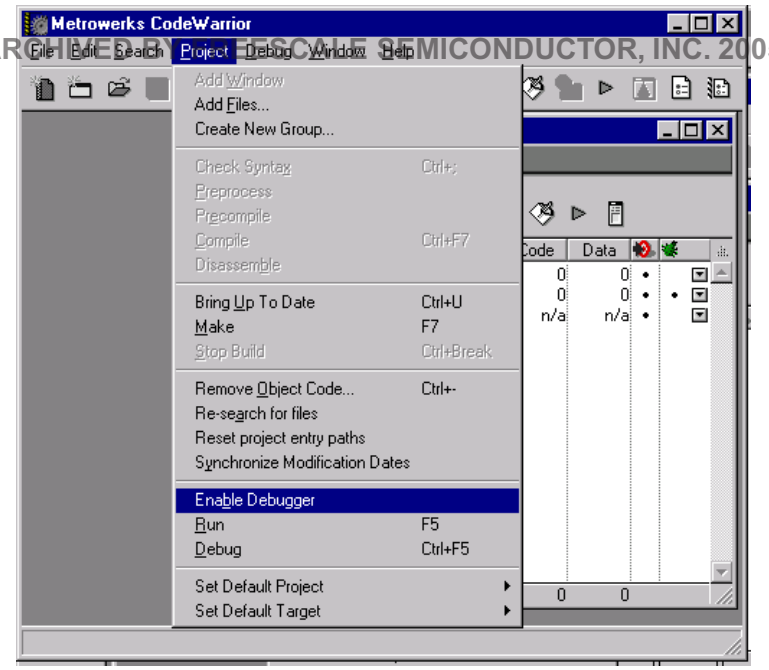
1. Install the Metrowerks Codewarrior IDE
2. Run Codewarrior IDE program.
3. Open a new project file with Embedded 68k Stationery.



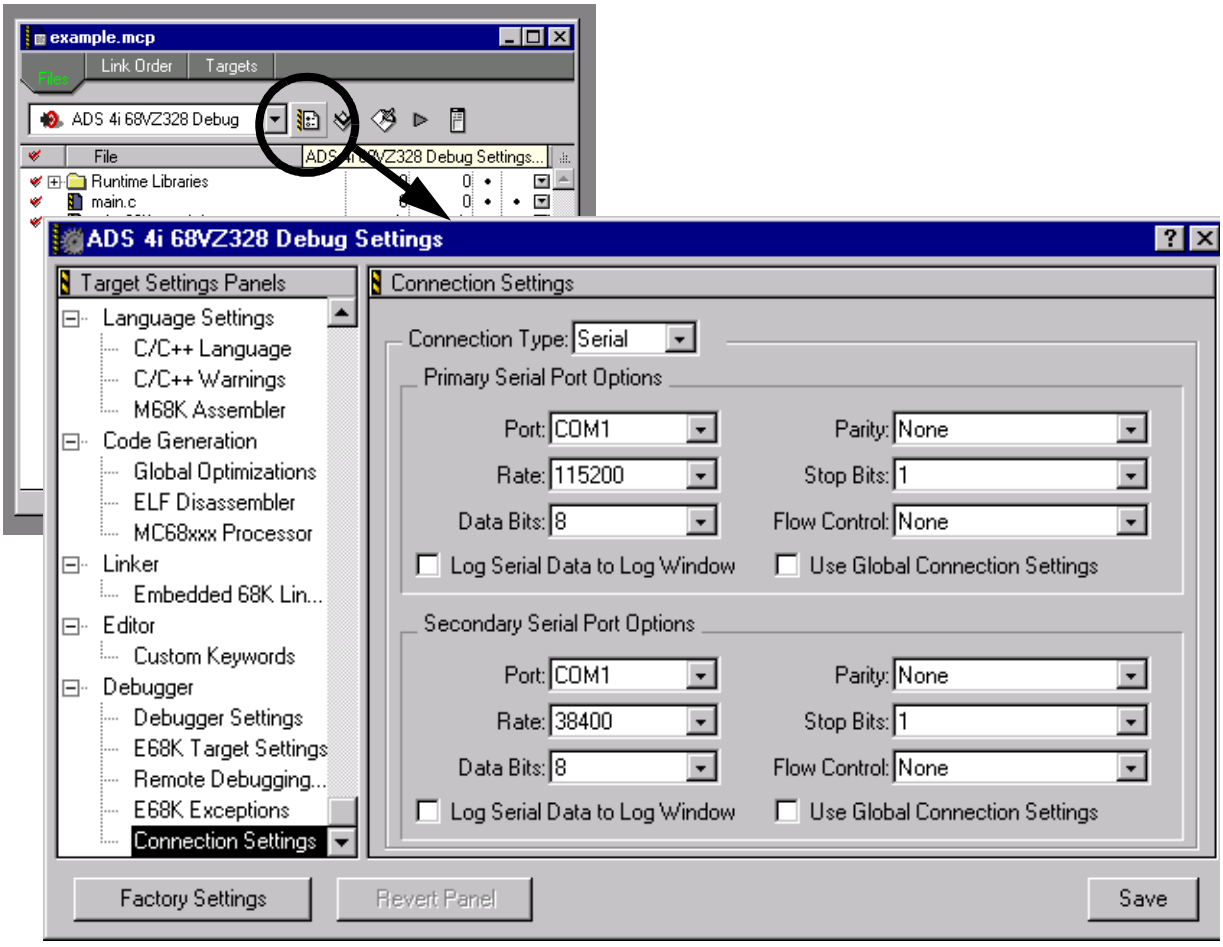
4. Select ADS_68EZ328 Stationery for new project.



5. Choose "Enable Debugger" from the "Project" pull-down menu.



6. Change the connection settings in the Debug Settings Windows.



7. Edit the code inside the Codewarrior IDE program.
8. Press F5 to run the program.

Single Step Development System

A simple procedure for using Single Step Debugger:

1. Install the Single Step Debugger on your PC.
2. Run Single Step Debugger
3. Choose **Debug** in the **File** pull-down menu to open the **Debug** pop-up window.
4. Inside the **Debug** pop-up window, choose the object file to download or "debug without file" as shown in Figure 2-5, select the serial port (COM1 or COM2 ...) according to the serial port of the PC connecting to the ADS, disable "hardware flow control" and the baud rate should be 115200bps as shown in Figure 2-6.
5. The file should be downloaded and then you can start your development. (For details, please refer to the SingleStep User's Manual).

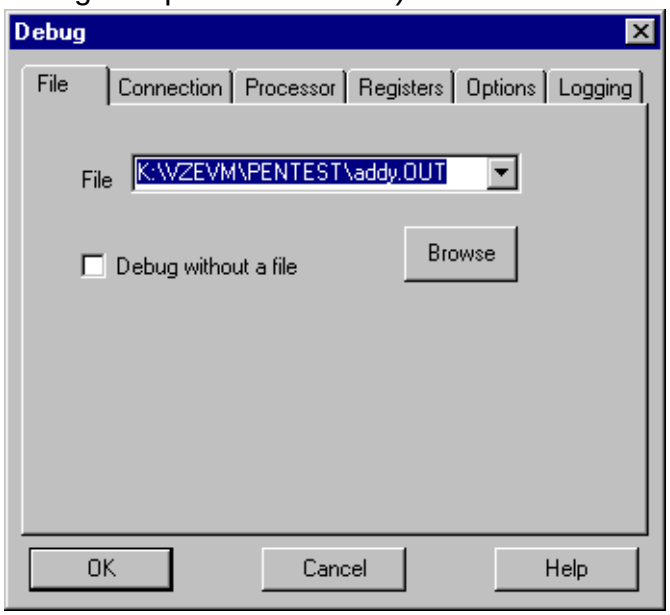


Figure 2-5. Debug Pop-Up Window of SDS v7.4

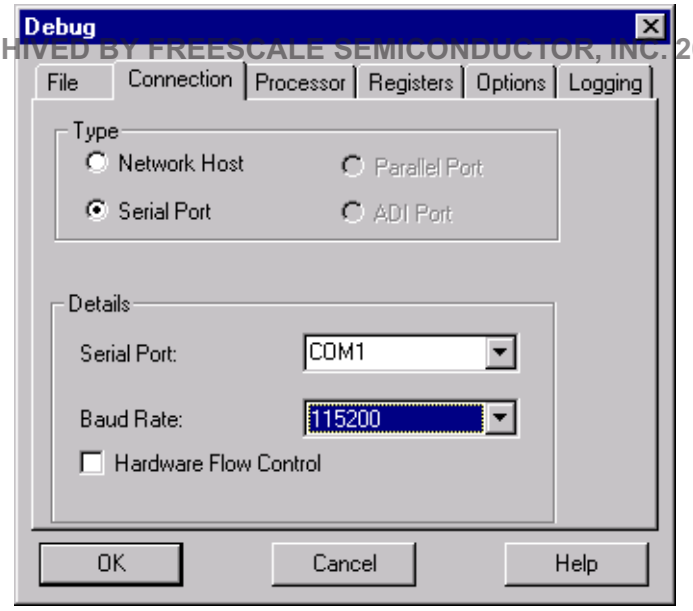


Figure 2-6. Connection Setting inside Debug Pop-up Window of SDS v7.4

SECTION 3
HARDWARE DESCRIPTION AND BOARD OPERATION

3.1 OVERVIEW

Figure 3-1 shows the block diagram of M68EZ328ADS v2.0.

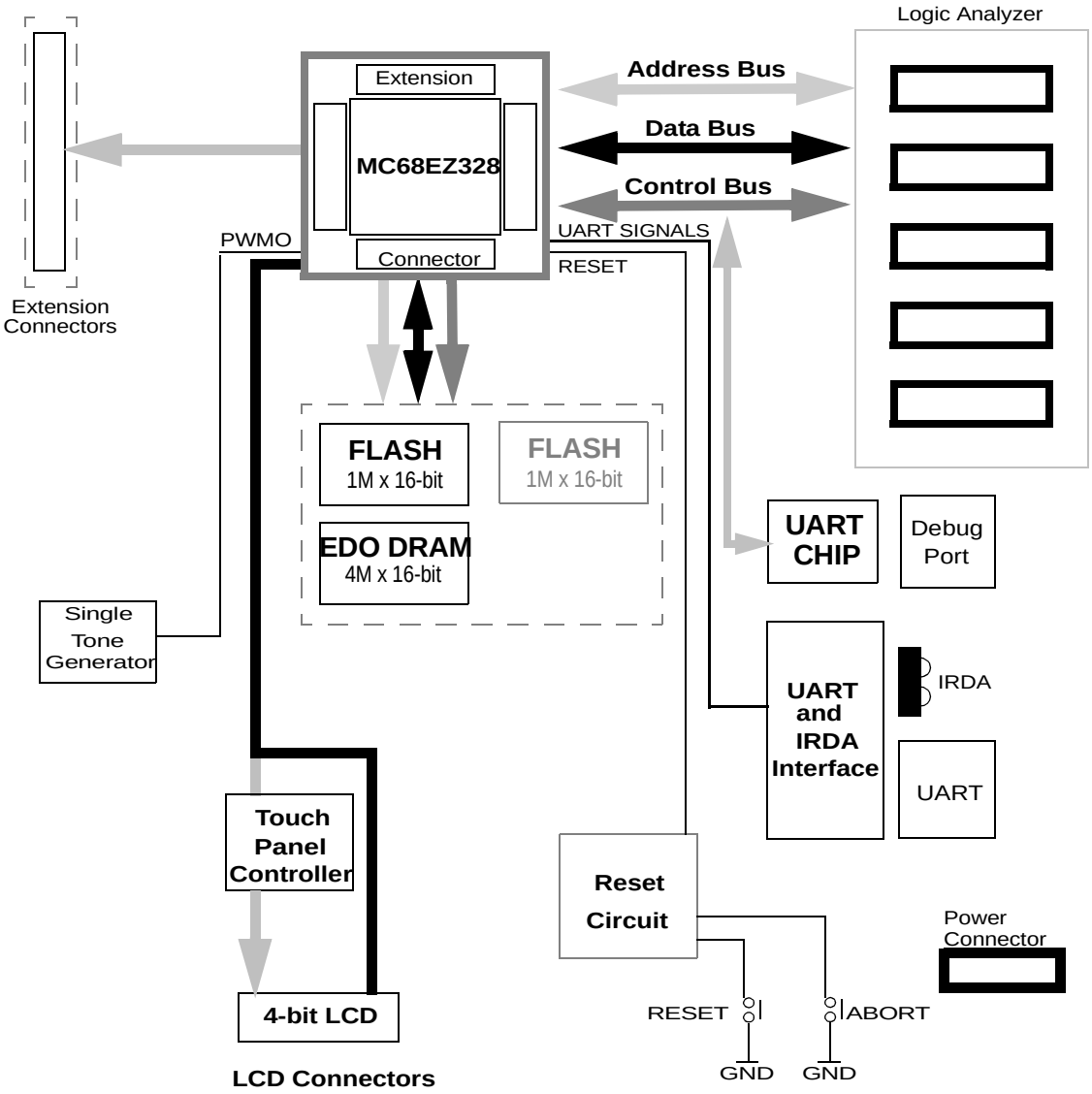


Figure 3-1. M68EZ328ADS v2.0 Functional Block Diagram

3.2 CONTROL SWITCHES

There are two push buttons on the ADS which function as follows:

1. **Reset Switch (SW2)**: When pressed, a hardware reset is generated to the MC68EZ328 processor and resumes operation.
2. **Abort Switch (SW1):** This switch is used to generate a level 7 interrupt to the MC68EZ328 processor for aborting normal software execution and returning control to the debug monitor.

3.3 DIP SWITCHES

There are two DIP switch packs on the ADS board, S1 and S2. S1 is used to configure the memory system and operation modes. S2 is used to enable the on-board peripheral such as buzzer and touch panel controller. Table 3-1 and Table 3-2 show the description of each switch.

Table 3-1. DIP Switch pack S1 Setting

Switch	FUNCTION	ON	OFF
S1-1	UART Transceiver	Enable	Disable
S1-2	IrDA Module	Enable	Disable
S1-3	Buzzer	Enable	Disable
S1-4	Pen IRQ	Enable	Disable
S1-5	Chip Select of Touch Panel Controller	Enable	Disable
S1-6	Unused		
S1-7	Unused		
S1-8	Unused		

Table 3-2. DIP Switch pack S2 Setting

Switch	FUNCTION	ON	OFF
S2-1	CSA0 - FLASH0	Connected	Disconnected
S2-2	CSA1 - FLASH1	Connected	Disconnected
S2-3	CSD0 - EDO DRAM	Connected	Disconnected
S2-4	EMU Mode	Enable	Disable
S2-5	Bootstrap Mode	Enable	Disable
S2-6	Monitor Select	SDS	Metrowerks
S2-7	Unused		
S2-8	Unused		

3.4 OPERATION MODES

M68EZ328ADS v2.0 supports three operation modes of MC68EZ328: EMU Mode, Normal Mode and Bootstrap Mode. Selection of these operation modes is controlled by DIP switch S2-8. Operation mode has to be selected before resetting the system. Mode is not allowed to change during normal running. Table 3-3 shows the operation mode configuration.

Table 3-3. Operation Mode Setting

DIP Switch S2-4	DIP Switch S2-5	Operation Mode
ON	OFF	EMU
OFF	OFF	Normal
ON	ON	Bootstrap
OFF	ON	Bootstrap

Normal Mode - After power up or system reset in this mode, CSA0 is default to cover the whole memory map except MC68EZ328 internal registers and EMU space. Also, as reset vector fetch is at the beginning of CSA0 space, CSA0 should be connected to the boot ROM in which the first two words are reset vectors.

Bootstrap Mode - When this mode is selected, the DragonBall-EZ will start its embedded bootloader. User can use this mode to do simple debugging or reprogram the flash memories. For detailed bootstrap mode operation, please refer MC68EZ328 user's manual.

EMU Mode - When this mode is selected, the reset vectors are generated internally by the ICE module of MC68EZ328. The first instruction fetch is at $FFFC0020. Therefore, in EMU mode the debug monitor of MC68EZ328ADS v2.0 is located beginning at $FFFC0020. EMUCS is always running in 8-bit data bus mode covering the address space from $FFFC0000 to $FFFD0000

3.5 LED INDICATORS

There are four LED indicators on the ADS which function as shown in Table 3-4.

Table 3-4. Function of LED Indicators

Reference #	Color	Name	Function
LED3	Yellow	PD0	Status of PD0
LED4	Yellow	PD1	Status of PD1
LED1	Red	Heart Beat	Blinking heart beat indicates the system is "alive"
LED2	Green	Power	Power is applied to the system with right polarity

The LED3 is connected to a counter. The counter is toggled by address line A1.

3.6 MEMORY

M68EZ328ADS v2.0 provides on-board Flash memory and EDO DRAM, for application development. They can be enabled or disabled individually by setting the corresponding DIP switches.

3.6.1 Memory Map

The default memory map of M68EZ328ADS v2.0 in normal mode is shown in Table 3-5. The chip select range to all of the memory are software programmable. Users can reconfigure the memory map for their applications.

Table 3-5. M68VZ328ADS v2.0 Default Memory Map

System Address	Memory	Assigned Chip Select
$00000000-$007FFFFF	8MB EDO DRAM	CSD0
$00800000-$009FFFFF	2MB FLASH BANK0	CSA0
$00A00000-$00BFFFFF	2MB FLASH BANK1	CSA1
$00C00000-$00FBFFFF	Unused	-
$FFFC0000-$FFFCFFFF	EMU Monitor	EMUCS
$FFFD0000-$FFFDFFFF	Debug Port	EMUCS
$FFFFF000-$FFFFFFFF	MC68EZ328 Internal Register	-

3.6.2 FLASH Memory

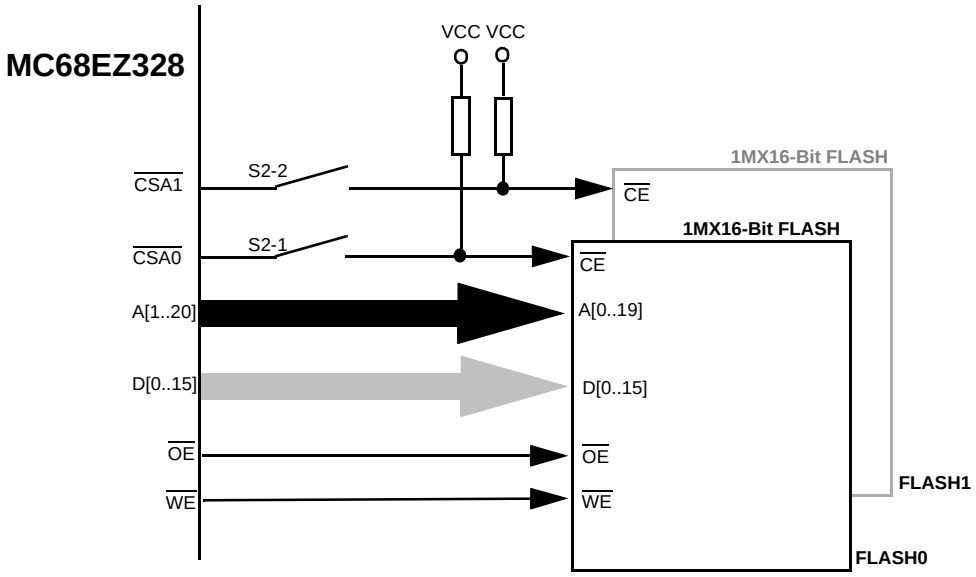


Figure 3-2. Interface of Flash Memories

M68EZ328ADS v2.0 supports up to two 2M-byte Flash memory chips. However, only one bank is installed when the board is shipped out from the factory. Figure 3-2. shows the interface of them. They are chip-selected by *CSA0 and *CSA1 signals. The connection of these *CSAx signals to the Flash memories is controlled by DIP switches S2-1 and S2-2.

When S2-1 is ON, *CSA0 is connected to FLASH0. When S2-2 is ON, *CSA1 is connected to FLASH1. In Normal Mode, FLASH0 is the boot ROM. User can reprogram Flashes with their own application program. For more details on flash memory programming, please refer to Appendix C .

3.6.3 EMU ROM

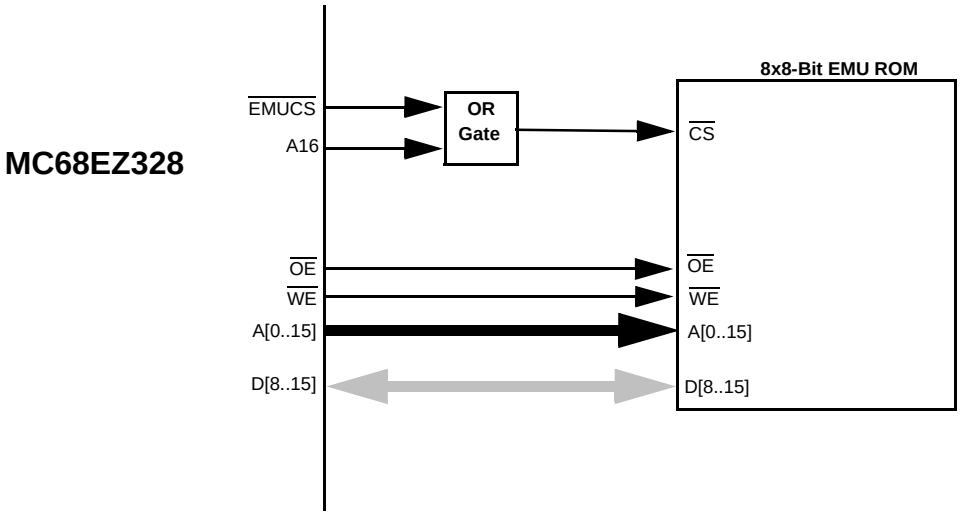


Figure 3-3. Interface of EMU ROM

The M68EZ328ADS v2.0 is equipped with one 64K-byte EMU ROM. Figure 3-3 shows the interface of the EMU ROM. This EMU ROM has already been programmed with the SDS monitor and the Metrowerks monitor when the board is shipped out from the factory.

3.6.4 EDO DRAM

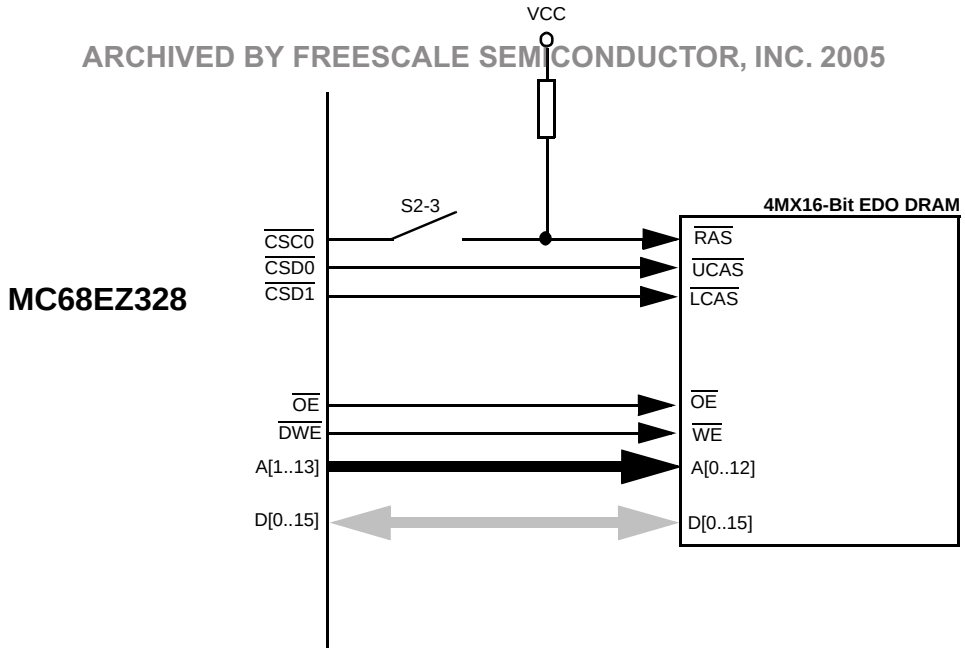


Figure 3-4. Interface of EDO DRAMs

Figure 3-4. shows the interface of EDO DRAMs. M68EZ328ADS v2.0 supports one bank of 4Mx16-bit EDO DRAMs. The EDO DRAM is enabled by turning DIP switch S2-3 on.

3.7 UART AND IRDA

MC68VZ328

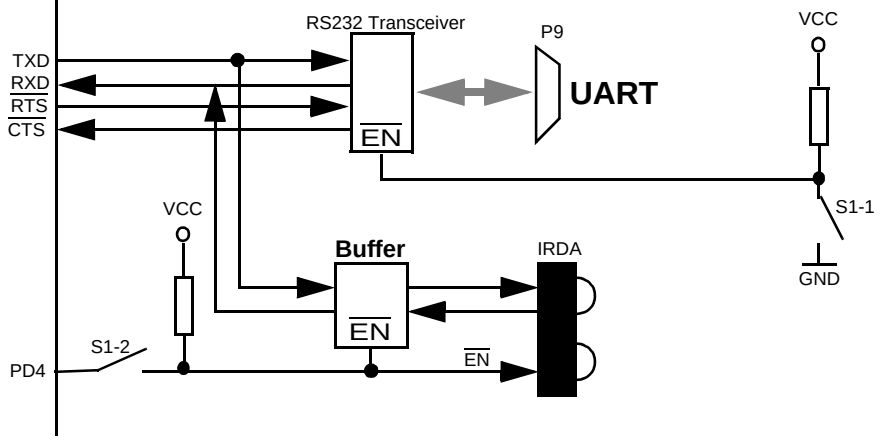


Figure 3-5. UART and IRDA Interface.

Figure 3-5 shows the UART and IRDA module. The M68EZ328ADS v2.0 has one RS232 serial ports P9. P9 is using the UART of MC68EZ328. It is 9-pin female D-Type connectors containing the signals as shown in Figure 3-6. The transceivers for UART can be enabled by turning the DIP switch S1-1 on.

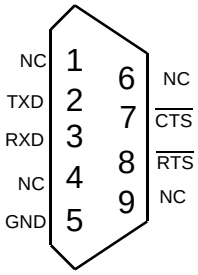


Figure 3-6. Serial Port Pin Assignment

The IrDA transceiver is provided and connected to DragonBall's UART Port only with a buffer in between for controlling its ON/OFF. If IrDA is being used, S1-1 should be switched OFF and S1-2 should be switched ON. Then, the enable/disable of IRDA is totally controlled by software through PD4 (Port D4).

3.8 DEBUG PORT

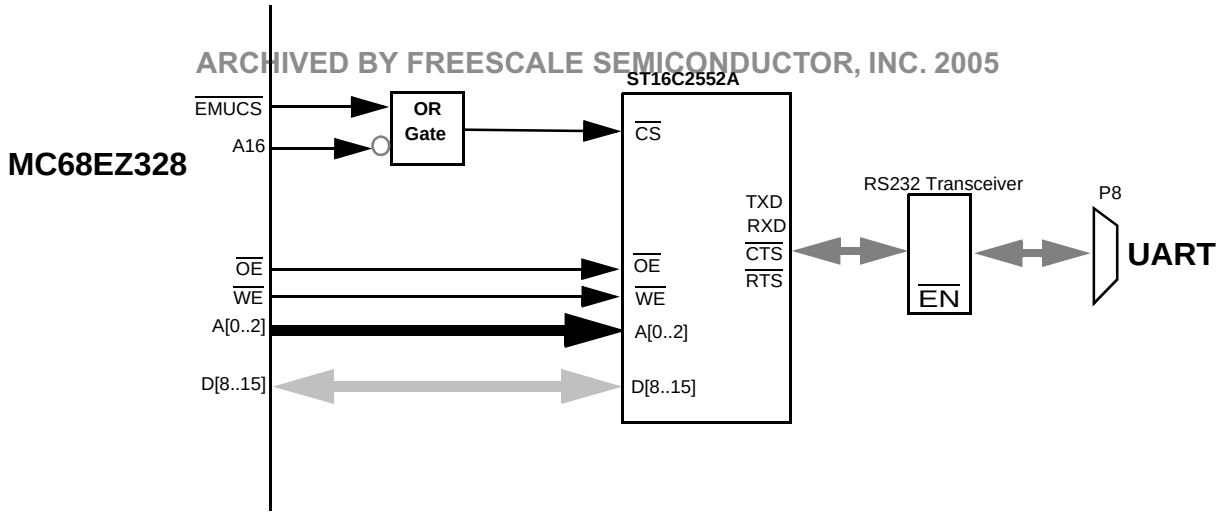


Figure 3-7. Debug Port Interface

As shown in Figure 3-7, an external UART chip ST16C2552A is used to provide one more debug port on the ADS board. This UART chip is connected with the DB9 connector P8. Its pin assignment is shown in Figure 3-6. The baud rate of this UART chip is defaulted to 115200bps.

3.9 LCD AND TOUCH PANEL INTERFACE

3.9.1 LCD Interface

M68EZ328ADS consists of one LCD panel connector, P10. P10 is designed for 4-bit LCD panel. The pin assignments of the LCD connector is shown in Figure 3-8.

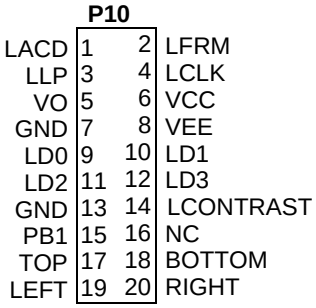


Figure 3-8. LCD and Touch Panel Connector Pin Assignment

For full description of the LCD signals, please refer to the MC68EZ328 User's Manual.

3.9.2 Touch Panel Interface

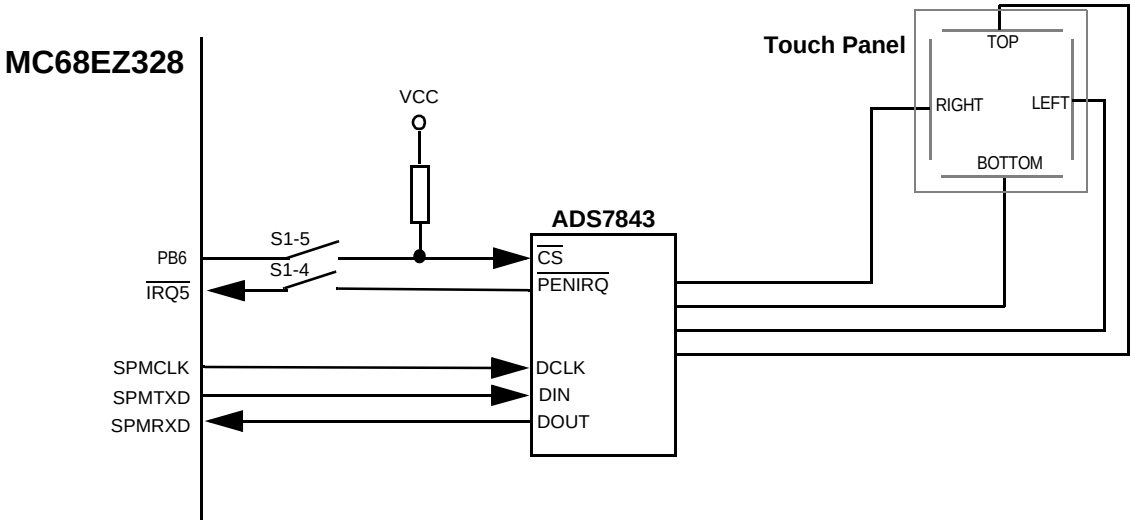


Figure 3-9. Touch Panel Controller Interface

The M68EZ328ADS board is equipped with Burr-Brown Touch Panel Controller ADS7843. Figure 3-9 shows the interface of Touch Panel Controller. Totally one I/O signal, one interrupt capable I/O signal and one set of SPIM signals from MC68EZ328 are used to implement the touch panel circuitry. All of these signals can be shared for other devices when touch panel interface controller is being disabled. The on-board touch panel controller can be disabled by turning both DIP switches S1-4, S1-5 off.

Table 3-6. Signal Assignment for Touch Panel Controller

Signals	Function
SPMRXD, SPMTXD, SPMCLK	This set of serial port interface signals performs data transfer with ADS7843. For detailed operation on the SPI port, please refer to the MC68EZ328 User's Manual.
~IRQ5	This signal is a MC68EZ328 interrupt capable I/O pin.
PB6	This MC68EZ328 general I/O signal is used to select the ADS7843 on data transfer.

The ADS7843 is a 12-bit sampling analog-to-digital converter (ADC) with a synchronous serial interface and low on-resistance switches for touch screens. The ADS7843 communicates with MC68EZ328 through SPI (Serial Peripheral Interface). In this case, ADS7843 is a slave, and MC68EZ328 is a master. For the detailed operation of ADS7843, please refer to its datasheet. The datasheet is available at the following web site.

http://www.burr-brown.com

For more details on the operation of resistive touch panel, please refer to Appendix B .

3.10 SINGLE TONE GENERATOR

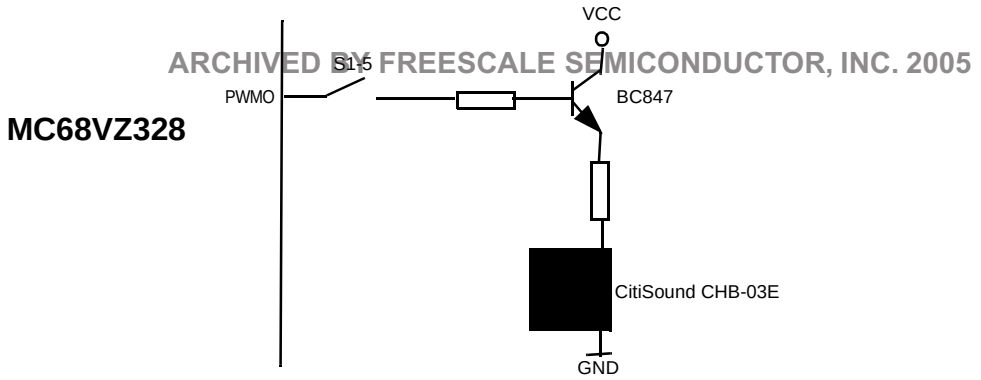


Figure 3-10. Single Tone Generator

Citizen single tone generator CHB-03E is provided on M68EZ328ADS v2.0. As shown in Figure 3-10, a simple transistor circuit is used to interface the CHB-03E with the PWMO pin of MC68EZ328.

3.11 LOGIC ANALYZER INTERFACE

To provide an convenient way of connecting on-board signals to a logic analyzer, ADS board provides five 10x2-pin headers for direct plug-in. Pin assignments to these five headers are shown in Figure 3-11.

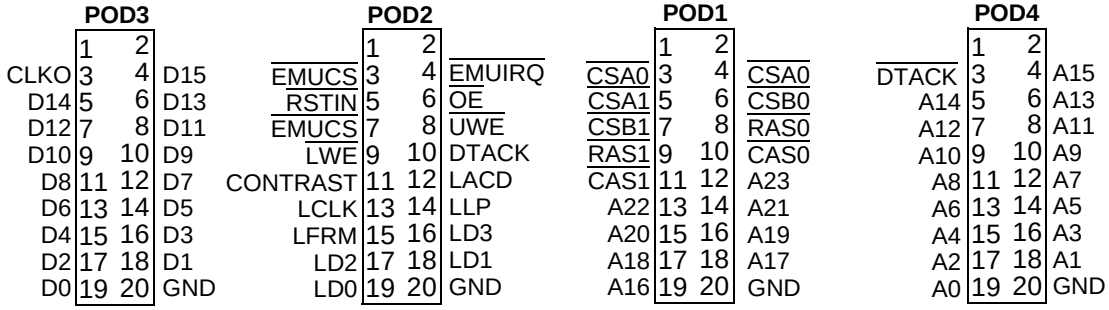


Figure 3-11. Logic Analyzer Connectors

3.12 EXPANSION CONNECTORS

The M68EZ328ADS v2.0 provides basic features for software development and evaluation. If user wants to add application subsystem to M68EZ328ADS v2.0, it can utilize the signals provided on 32x3 local bus connector (P4). Some of these MC68EZ328 signals are used by the on-board modules. If users want to use any of these signals for their daughter card, the corresponding on-board module may have to be disabled to avoid contention.

The pin assignments for extension connectors are shown in Table 3-7.

Table 3-7. Pin Assignments of P4

Pin#	A	B	C
1	GND	GND	GND
2	D0/PA0	D1/PA1	D2/PA2
3	D3/PA3	D4/PA4	D5/PA5
4	D6/PA6	D7/PA7	D8
5	D9	D10	D11
6	D12	D13	D14
7	D15	CSA0	PF7/CSA1
8	PB0/CSB0	PB1/CSB1	PB2/CSC0/RAS0
9	PB3/CSC1/RAS1	PB4/CSD0/CAS0	PB5/CSD1/CAS1
10	PB6/TOUT/TIN	PB7/PWMO1	PC0/LD0
11	PC1/LD1	PC2/LD2	PC3/LD3
12	PC4/LFRM	PC5/LLP	PC6/LCLK
13	PC7/LACD	PD0/INT0	PD1/INT1
14	PD2/INT2	PD3/INT3	PD4/IRQ1
15	PD5/IRQ2	PD6/IRQ3	PD7/IRQ6
16	PE0/SPMTXD	PE1/SPMRXD	PE2/SPMCLK
17	PE3/DWE/UCLK	PE4/RXD	PE5/TXD
18	PE6/RTS	PE7/CTS	PF0/LCONTRAST
19	PF1/IRQ5	PF2/CLKO	PG1/A0
20	MA0/A1	MA1/A2	MA2/A3
21	MA3/A4	MA4/A5	MA5/A6
22	MA6/A7	MA7/A8	MA8/A9
23	MA9/A10	MA10/A11	MA11/A12
24	MA12/A13	MA13/A14	MA14/A15
25	MA15/A16	A17	A18
26	A19	PF3/A20	PF4/A21
27	PF5/A22	PF6/A23	PG2/EMUIRQ
28	PG3/HIZ/P/D	PG4/EMUCS	PG5/EMUBRK
29	PG0/BUSW/DTACK	RESET	OE
30	UWE	LWE	PK2/LDS
31	PK3/UDS	PK1/R/W	NC
32	VCC	VCC	VCC

3.13 POWER SUPPLY

There are two power input connectors on the ADS, P1 and P2. P1 is designed for external 3V DC main power supply, which supplies power to the MC68EZ328 processor and most of the on-board components. P2 is used for LCD driver (VEE) power supply. If an LCD panel

is connected to the ADS, the LCD driver can be powered by this VEE input. Figure 3-12 locates the power connectors and their polarity.

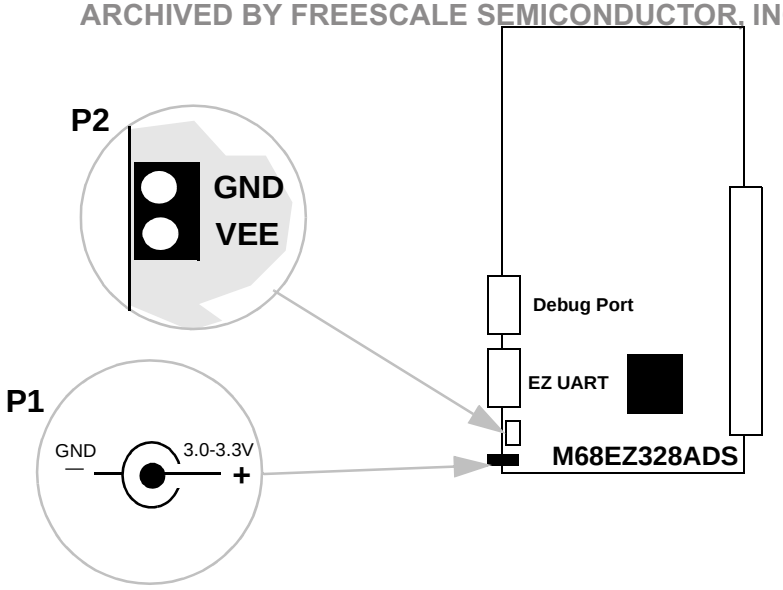


Figure 3-12. Power Connectors

APPENDIX A
COMPARISON OF M68EZ328ADS VERSION 1.X AND 2.0

This section describes the difference between the M68EZ328ADS version 1.x and 2.0. If user is not upgrading the development platform from M68EZ328ADS v1.x to v2.0, this section can be ignored.

A.1 HARDWARE

Table A-1 shows the difference between the M68EZ328ADS v1.x and v2.0.

Table A-1. Comparison of M68EZ328ADS v1.x and v2.0

Descriptions	Version 1.x	Version 2.0
Flash Memory	2M-byte Flash Memory using four AT29BV040	2M-byte Flash Memory using one MBM29LV160T (expandable to 4M-byte Flash Memory)
SRAM	256K-byte SRAM	No SRAM
EDO DRAM	4M-byte EDO DRAM using the $\overline{WE}$ signal of MC68EZ328	8M-byte EDO DRAM using the $\overline{DWE}$ signal of MC68EZ328
Touch Panel Interface	MAXIM MAX1249 and four transistors	Burr-Brown ADS7843E
Expansion Socket	Same pin assignment	Same pin assignment
LCD Connector	Same pin assignment	Same pin assignment
IrDA	No	HSDL-3201
Debug Port	ADI Port and MC68681 Serial Port with max. baud rate = 57600bps	ST16C2552A Serial Port with max. baud rate = 115200bps
Buzzer	No	Citizen CHB-03E
Bus Width	8-bit/16-bit	16-bit only

A.2 SOFTWARE

Table A-2 shows the software change in M68EZ328ADS v2.0.

Table A-2. Software Change for using M68EZ328ADS v2.0

Items	Descriptions
Initialization Code	The new ADS is equipped with different memory device. So, the initialization code needs to be changed.
Touch Panel Controller Device Driver	Burr-Brown ADS7843E is used. A new device driver is required.
$\overline{DWE}$	Pin 12 ($\overline{DWE}$/UCLK/PE3) of MC68EZ328 is programmed as DWE signals for on-board EDO DRAM. It cannot be used as Port or UCLK again.
IrDA	The IrDA module needs to be enabled by setting PD4 to zero.

APPENDIX B
RESISTIVE TOUCH PANEL OPERATION

M68VZ328ADS features pen input through a resistive-film sensing panel. This type of panel provides high flexibility by accepting input form any kind of stimulus including fingers, which is most suitable for portable use. User can select a specific touch panel or order an LCD module which includes a touch resistive panel. This section describes the basic concepts of pen input and the required interface with the M68VZ328ADS.

B.1 GENERAL CONCEPTS OF RESISTIVE PANELS

Basically, resistive panel consists of two transparent resistive layers separated by insulating spacers as shown in Figure 2-1.

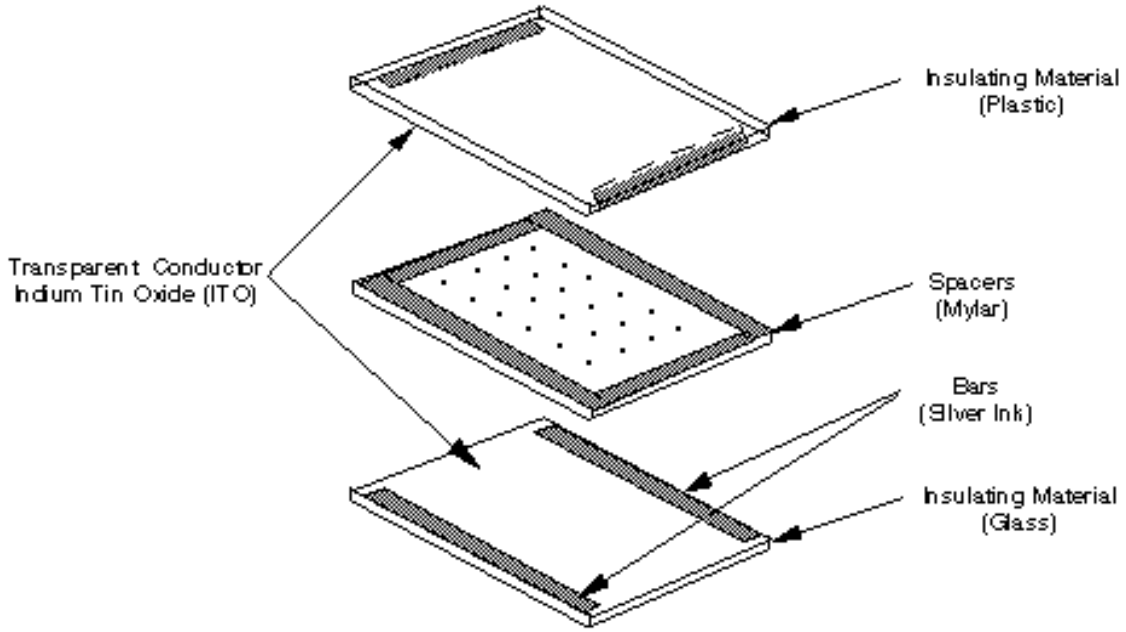


Figure 2-1. Resistive Touch Panel

Transparent Resistive Layer - resistive material such as an indium tin oxide (ITO) film is coated on dielectric (insulating) substrate, usually glass on bottom and plastic on top for actuation.

Bars - highly conductive material such as silver ink, about 1000 times more conductive than ITO.

Spacers - Non-conducting ink, adhesive, or other material such as Mylar is used to separate the two opposite conductive layers.

The resistive panel works by applying a voltage gradient across one conductive layer and measuring the voltage at the point of contact with the opposing conductive layer. For instance, as shown in Figure 2-2, the resistive film acts as a series of resistors.

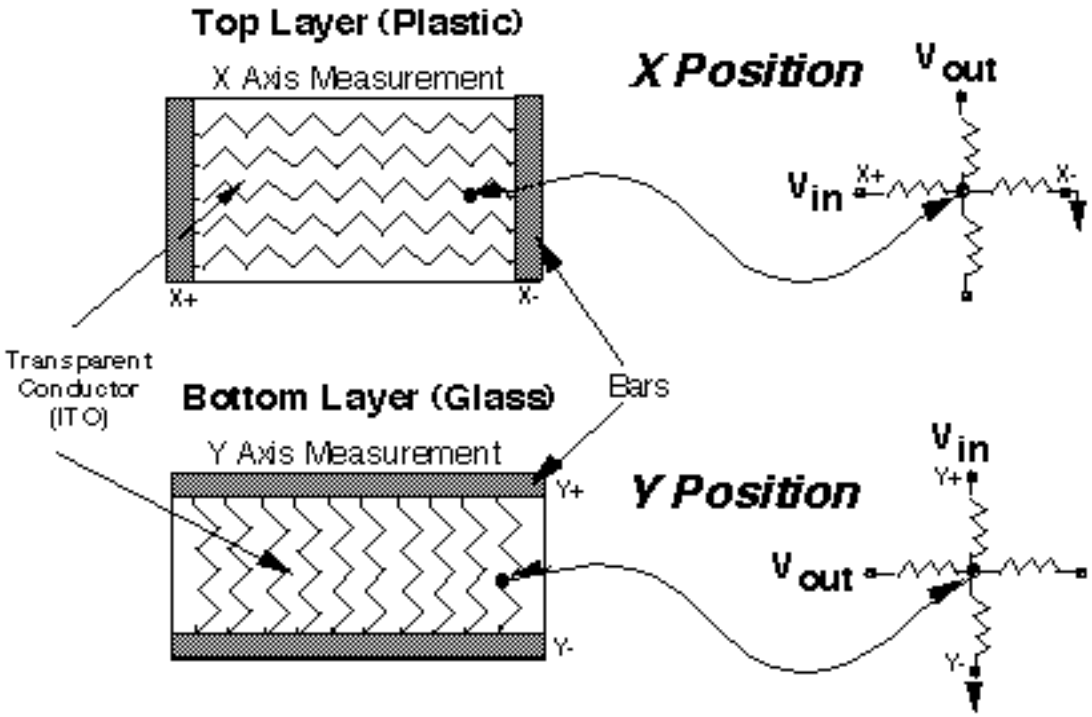


Figure 2-2. Determination of X,Y Position

When a point is contacted, it means the two opposing conductive layers come into electrical contact. The x position of actuation can be determined by measuring the output voltage of the y layer. At the same time, the y position can be find out by measuring the x layer. The exact position can be determined by referencing the output voltage to the distance relationship.

APPENDIX C
PROGRAMMING ON-BOARD FLASH MEMORY

C.1 OVERVIEW

The Flash memory on the ADS board cannot be written directly. A special program command sequence is required to unlock it before starting the write process. A flash program is therefore provided with the ADS board for helping users to do re-programming. The sections below will describe the program and provide further information about the process and other required elements.

It is recommended that user should have a basic understanding of bootstrap mode operation of MC68EZ328 before reading the material below. For more details on Bootstrap mode, please refer to the MC68EZ328 user's manual.

C.2 ELEMENTS FOR PROGRAMMING THE FLASH

The following files are necessary for programming the Flash memory

1. EZTOOLS including BBUG.EXE and STOB.EXE
2. INIT.B - b-record for initializing the ADS
3. ERASE.B - b-record for erasing the flash memory
4. FLASHNML.B - b-record for programming flash. ROM image is copied from RAM area to the Flash memory area
5. ROM.B - ROM image of user program in b-record/s-record format.

C.3 METHOD

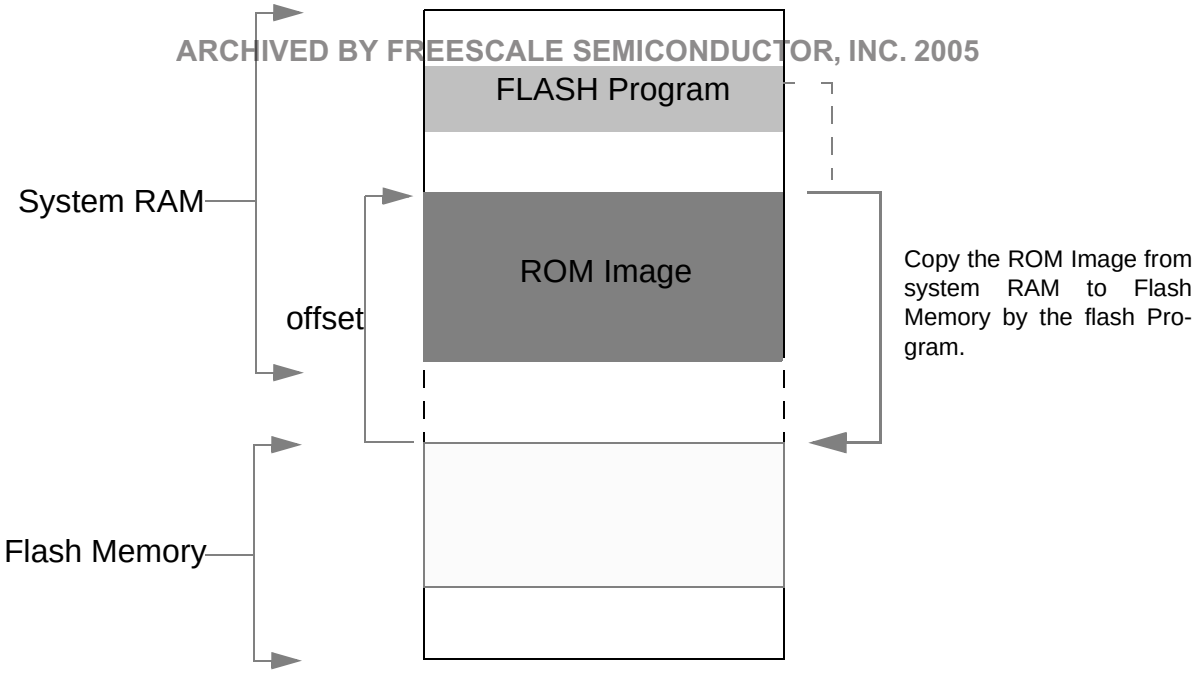


Figure C-1. Method of programming on-board Flash memory

Flash memory can be programmed in bootstrap mode. First, a ROM image and a flash program are downloaded to the system memory by loading their b-records. Then, run the flash program. It will execute the Flash program command sequence and copy the ROM image from the system RAM to the Flash memory. The detailed steps are as follows.

1. Force the MC68EZ328 into bootstrap mode by turning S2-8 on and pressing RESET switch once.
2. Use BBUG.EXE or TERMINAL program to communicate with the M68EZ328ADS through RS232 port of a PC.
3. Initialize the internal registers of DragonBall-VZ by loading the INIT.B.
4. Make sure the Flash Memory is blank. Load ERASE.B to erase the flash memory when needed.
5. Load FLASHNML.B (the Flash Program) and ROM.B (the ROM image) to system RAM (EDO DRAM on M68EZ328ADS).
6. Execute the Flash Program by using the execution B-record. For example, if the starting address of Flash Program is 0x4000, the execution B-record is then "0000400000".

C.4 OFFSET ADDRESS OF ROM IMAGE

Figure C-1 shows that the ROM image is first put to the system RAM before it is copied to Flash. In order to create S-record/B-record with download address different from its execution address, an OFFSET is sometimes required to be specified in downloading program.

For example, when using SDS's DOWN.EXE to generate the s-record, the "-w offset" parameter can be used to specify this offset value. Please refer to the SingleStep User Guide for using this command.

C.5 EXECUTING PROGRAM COMMAND SEQUENCE

Listed below is the source code of the flash Program which contains the necessary steps to write the flash memory MBM29LV160T. It executes the flash program command sequence and copies the ROM image from RAM area to the Flash memory area on ADS board. Figure C-2 shows the flow chart of this program. Different brands of Flash memory may have different program command sequences, please refer to their datasheets for more details.

```
                XREF  STKTOP

OFFSET1         equ $AAA
OFFSET2         equ $554

TIME            equ $FFF

*****************************************************************
ECHO    MACRO   CHAR
        bsr     TXD_RDY
        nop
        nop
        nop
        move.b  #CHAR,$FFFFF907
        ENDM

*****************************************************************
ENABLE  MACRO
        move.w  #$00AA,(A5)  ; Unlock Flash
        move.w  #$0055,(A6)
        move.w  #$00A0,(A5)
        ENDM

*****************************************************************
        SECTION parameter
pSOURCE         DC.L   $00010000
pTARGET         DC.L   $01000000
pSIZE           DC.L   $00010000
pFLASH          DC.L   $01000000

pERROR          DC.L   $0
pFINISH         DC.L   $0
pERROR_ADDRESS  DC.L   $0
                DC.L   STKTOP
*****************************************************************
        SECTION code
START
       MOVEA.L #STKTOP,a7     ;re-istall stack pointer in case bootstrap mode

        MOVE.L  #0,pERROR
        MOVE.L  #0,pFINISH
        move.l  #0,pERROR_ADDRESS
```

```
        move.l  pSOURCE,A0
        move.l  pTARGET,A1
        move.l  pSIZE,D0
```

```
        move.l  pFLASH,A5
        move.l  pFLASH,A6

*****************************************
* BreakPoint Here if program flash in SDS *
*****************************************

        add.l   #OFFSET1,A5
        add.l   #OFFSET2,A6

***********************************************

        ;***************************************
        ; Input Parameters:
        ;     a0 - (Long) Source Address
        ;     a1 - (Long) Target Address
        ;     d0 - (Long) Byte Size
        ;     a5 - (long) Flash Starting Address Offset1
        ;     a6 - (long) Flash Starting Address Offset2
        ;***************************************

        move.l  a0,a2           ; a2 as source
        move.l  a1,a3           ; a3 as target

        clr.l     d1            ; clear d1 as counter
        clr.b     d5            ; clear d5 as ECHO counter

PROGRAM
        ENABLE
        move.w    (a2),(a3)     ; Copy source to target

        clr.l     d4            ; clear d4 as polling counter
POLLING
        cmp.l     #TIME,d4
        bgt       ERROR         ; if d4>#TIME, bra ERROR
        add.l     #1,d4         ; d4++
        move.w    (a2),d2       ; Compare (a0),(a1)
        move.w    (a3),d3
        cmp.w     d2,d3
        bne       POLLING       ; End of POLLING

        add.l     #2,a2
        add.l     #2,a3
        add.l     #2,d1

        cmp.b     #0,d5
        bne       NO_W
        ECHO      'W'
        ECHO      0
NO_W    add.b     #1,d5
```

```
        cmp.l     d0,d1       ; if d1<=d0, bra PROGRAM
        ble       PROGRAM
```



```
        ;***************************************
        ; Input Parameters:
        ;     a0 - (Long) Source Address
        ;     a1 - (Long) Target Address
        ;     d0 - (Long) Byte Size
        ;     a5 - (long) Flash Starting Address Offset 1
        ;     a6 - (long) Flash Starting Address Offset 2
        ;***************************************
       move.l  a0,a2          ;a2 & a3 are used to store the starting address
        move.l  a1,a3           ;for comparing.

        clr.b   d5              ;clear d5 as ECHO counter
        clr.l   d1              ;clear d1 as counter
VERIFIY
        cmp.b     #0,d5
        bne       NO_V
        ECHO      'V'
        ECHO      0
NO_V    add.b     #1,d5

        move.w    (a2)+,d2
        move.w    (a3)+,d3
        cmp.w     d2,d3
        bne       ERROR

        add.l     #2,d1

        cmp.l     d0,d1         ; if d1<=d0, bra CHECK
        ble       VERIFIY

        bra    FINISH

*******************************************************************************

TXD_RDY
        move.l  d7,-(a7)      ; Push d7
POLLTXD
        btst.b  #5,$FFFFF906
        beq.s   POLLTXD       ; bra if TXAVAIL not SET
        move.w  #$0FFF,d7
LP_RDY  sub.w   #1,d7
        bne.b   LP_RDY
        move.l  (a7)+,d7      ; Restore d7
        rts

*******************************************************************************

FINISH
        ECHO    '\n'
        ECHO    'P'
        ECHO    'A'
```

```
        ECHO    'S'
        ECHO    'S'
        ECHO    '\n'
        ECHO    0
        ECHO    0
        ECHO    0
        ECHO    0
        ECHO    0
        ECHO    0
        move.l  #1,pFINISH
        bra     BOOTSTRAP

*************************************************************

ERROR
        ECHO    '\n'
        ECHO    'E'
        ECHO    'R'
        ECHO    'R'
        ECHO    'O'
        ECHO    'R'
        ECHO    '\n'
        ECHO    0
        ECHO    0
        ECHO    0
        ECHO    0
        ECHO    0
        ECHO    0
        sub.l   #1,a3
        move.l  a3,pERROR_ADDRESS
        move.l  #1,pERROR
        bra     BOOTSTRAP

*************************************************************

BOOTSTRAP
        jmp     $FFFFFF44

*************************************************************
        END
```

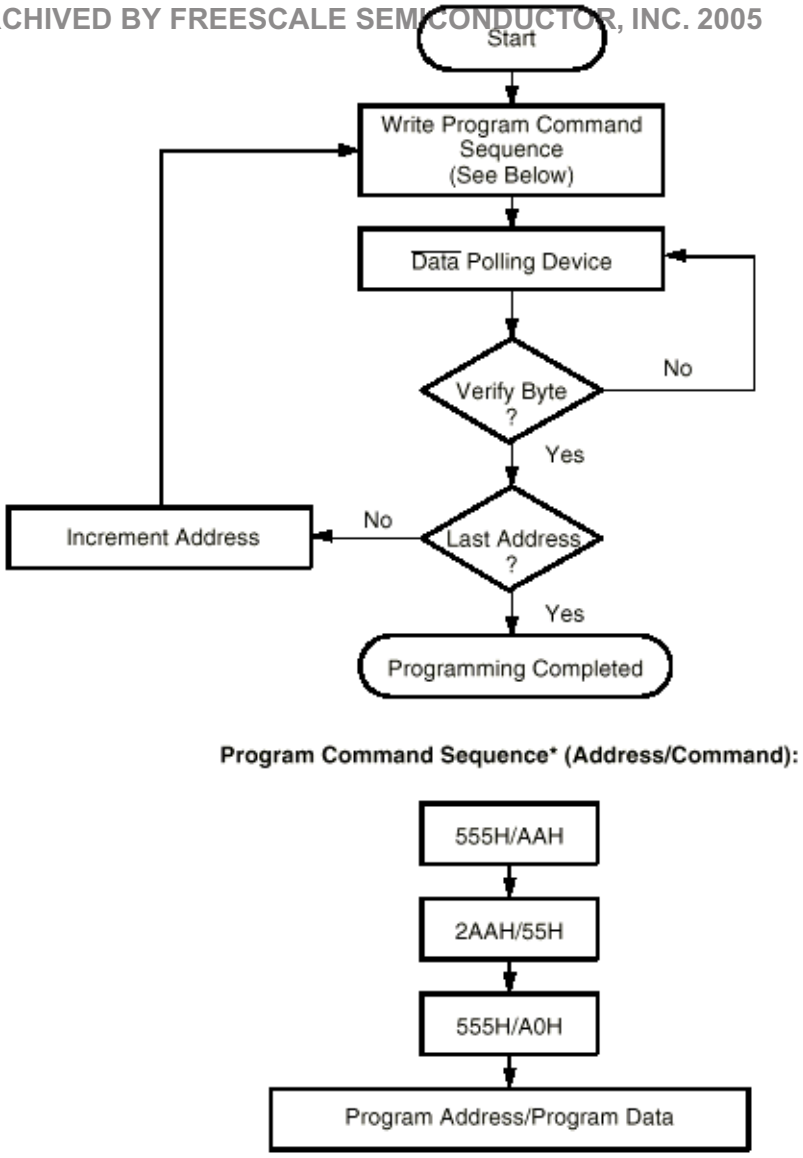


Figure C-2. Flash Program Algorithm

APPENDIX D
MONITOR INITIALIZATION CODE

D.1 INITIALIZATION CODE OF METROWERKS MONITOR (RESET.S)

```
    MON_STACKTOP.equ$4100;  Above is TOO low, try this

    M328BASE        .equ $FFFFF000

    ; SIM28 System Configuration Registers
    SCR             .equ     (M328BASE+$000)

    ; Chip Select Registers
    GRPBASEA        .equ     (M328BASE+$100)
    GRPBASEB        .equ     (M328BASE+$102)
    GRPBASEC        .equ     (M328BASE+$104)
    GRPBASED        .equ     (M328BASE+$106)
    CSA             .equ     (M328BASE+$110)
    CSB             .equ     (M328BASE+$112)
    CSC             .equ     (M328BASE+$114)
    CSD             .equ     (M328BASE+$116)
    DRAMCFG         .equ     (M328BASE+$C00)
    DRAMCTL         .equ     (M328BASE+$C02)
    EMUCS           .equ     (M328BASE+$118)
    CSCTR           .equ     (M328BASE+$150)

    ; PLL Registers
    PLLCR           .equ     (M328BASE+$200) ; Control Reg
    PLLFSR          .equ     (M328BASE+$202) ; Freq Select Reg
    PLLTSR          .equ     (M328BASE+$204) ; Test Reg

    ; Power Control Registers
    PCTLR           .equ     (M328BASE+$206) ; Control Reg

    ; Interrupt Registers
    IVR             .equ     (M328BASE+$300) ; Interrupt Vector Reg
    ICR             .equ     (M328BASE+$302) ; Interrupt Control Reg
    IMR             .equ     (M328BASE+$304) ; Interrupt Mask Reg
    ISR             .equ     (M328BASE+$30C) ; Interrupt Status Reg
    IPR             .equ     (M328BASE+$310) ; Interrupt Pending Reg

    ; PIO Registers
                             ; Port A Registers
    PADIR           .equ     (M328BASE+$400) ; Direction Reg
    PADATA          .equ     (M328BASE+$401) ; Data Reg
    PAPUEN          .equ     (M328BASE+$402) ; Pullup Enable Reg
                             ; Port B Registers
    PBDIR           .equ     (M328BASE+$408) ; Direction Reg
    PBDATA          .equ     (M328BASE+$409) ; Data Reg
    PBPUEN          .equ     (M328BASE+$40A) ; Pullup Enable Reg
    PBSEL           .equ     (M328BASE+$40B) ; Select Reg
                             ; Port C Registers
    PCDIR           .equ     (M328BASE+$410) ; Direction Reg
    PCDATA          .equ     (M328BASE+$411) ; Data Reg
```

```
PCPDEN          .equ    (M328BASE+$412) ; Pull-down Enable Reg
PCSEL           .equ    (M328BASE+$413) ; Select Reg
                         ; Port D Registers
PDDIR           .equ    (M328BASE+$418) ; Direction Reg
PDDATA          .equ    (M328BASE+$419) ; Data Reg
PDPUEN          .equ    (M328BASE+$41A) ; Pullup Enable Reg
PDSEL           .equ    (M328BASE+$41B) ; port D select
PDPOL           .equ    (M328BASE+$41C) ; Polarity Reg
PDIRQEN         .equ    (M328BASE+$41D) ; IRQ Enable Reg
PDIRQEDGE       .equ    (M328BASE+$41F) ; IRQ Edge Reg
                         ; Port E Registers
PEDIR           .equ    (M328BASE+$420) ; Direction Reg
PEDATA          .equ    (M328BASE+$421) ; Data Reg
PEPUEN          .equ    (M328BASE+$422) ; Pullup Enable Reg
PESEL           .equ    (M328BASE+$423) ; Select Reg
                         ; Port F Registers
PFDIR           .equ    (M328BASE+$428) ; Direction Reg
PFDATA          .equ    (M328BASE+$429) ; Data Reg
PFPUEN          .equ    (M328BASE+$42A) ; Pullup Enable Reg
PFSEL           .equ    (M328BASE+$42B) ; Select Reg
                         ; Port G Registers
PGDIR           .equ    (M328BASE+$430) ; Direction Reg
PGDATA          .equ    (M328BASE+$431) ; Data Reg
PGPUEN          .equ    (M328BASE+$432) ; Pullup Enable Reg
PGSEL           .equ    (M328BASE+$433) ; Select Reg

; PWM Registers
PWMC            .equ    (M328BASE+$500) ; Control Reg
PWMS            .equ    (M328BASE+$502) ; Sample Reg
PWMCNT          .equ    (M328BASE+$504) ; Count Reg

; Timer Registers
                         ; Timer 1 Registers
TCTL            .equ    (M328BASE+$600) ; Control Reg
TPRER           .equ    (M328BASE+$602) ; Prescalar Reg
TCMP            .equ    (M328BASE+$604) ; Compare Reg
TCR             .equ    (M328BASE+$606) ; Capture Reg
TCN             .equ    (M328BASE+$608) ; Counter
TSTAT           .equ    (M328BASE+$60A) ; Status Reg

; SPI Registers
SPIMDATA        .equ    (M328BASE+$800) ; Control/Status Reg
SPIMCONT        .equ    (M328BASE+$802) ; Data Reg

; UART Registers
USTCNT          .equ    (M328BASE+$900) ; Status Control Reg
UBAUD           .equ    (M328BASE+$902) ; Baud Control Reg
UARTRX          .equ    (M328BASE+$904) ; Rx Reg
UARTTX          .equ    (M328BASE+$906) ; Tx Reg
UARTMISC        .equ    (M328BASE+$908) ; Misc Reg
UARTNIPR        .equ    (M328BASE+$90A) ; None-Integer Prscaler reg

; LCDC Registers
LSSA            .equ    (M328BASE+$A00) ; Screen Start Addr Reg
LVPW            .equ    (M328BASE+$A05) ; Virtual Page Width Reg
```

```
LXMAX           .equ    (M328BASE+$A08) ; Screen Width Reg
LYMAX           .equ    (M328BASE+$A0A) ; Screen Height Reg
LCXP            .equ    (M328BASE+$A18) ; Cursor X Position
LCYP            .equ    (M328BASE+$A1A) ; Cursor Y Position
LCWCH           .equ    (M328BASE+$A1C) ; Cursor Width & Height Reg
LBLKC           .equ    (M328BASE+$A1F) ; Blink Control Reg
LPICF           .equ    (M328BASE+$A20) ; Panel Interface Config Reg
LPOLCF          .equ    (M328BASE+$A21) ; Polarity Config Reg
LACDRC          .equ    (M328BASE+$A23) ; ACD (M) Rate Control Reg
LPXCD           .equ    (M328BASE+$A25) ; Pixel Clock Divider Reg
LCKCON          .equ    (M328BASE+$A27) ; Clocking Control Reg
LRRA            .equ    (M328BASE+$A29) ; Refresh Rate Adjust reg
LPOSR           .equ    (M328BASE+$A2D) ; Panning Offset Reg
LFRCM           .equ    (M328BASE+$A31) ; Frame Rate Control Mod Reg
LGPMR           .equ    (M328BASE+$A33) ; Gray Palette Mapping Reg
LPWM            .equ    (M328BASE+$A36) ; contrast control Reg

; RTC Registers
RTCHMSR         .equ    (M328BASE+$B00) ; Hrs Mins Secs Reg
RTCALM0R        .equ    (M328BASE+$B04) ; Alarm Register
RTCDAY          .equ    (M328BASE+$B08) ; RTC date reg
RTCWD           .equ    (M328BASE+$B0A) ; RTC watch dog timer reg
RTCCTL          .equ    (M328BASE+$B0C) ; Control Reg
RTCISR          .equ    (M328BASE+$B0E) ; Interrupt Status Reg
RTCIENR         .equ    (M328BASE+$B10) ; Interrupt Enable Reg
RSTPWCH         .equ    (M328BASE+$B12) ; Stopwatch Minutes

;ICEM registers
ICEMACR         .equ    (M328BASE+$D00)
ICEMAMR         .equ    (M328BASE+$D04)
ICEMCCR         .equ    (M328BASE+$D08)
ICEMCMR         .equ    (M328BASE+$D0A)
ICEMCR          .equ    (M328BASE+$D0C)
ICEMSR          .equ    (M328BASE+$D0E)


*************************************************************************
*       RESET OPTIONS
*************************************************************************

        .section .reset
rom_base:
;--    SECTIONrom_reset - SP, start addr & space for Exception Vectors

        .DC.LMON_STACKTOP; stack pointer
        .DC.Lrom_start     ; program counter

        .org0x10                ; int7 handler (in EMU mode) goes here
        .externemu_lvl7_handler
        jmp    emu_lvl7_handler

        .org0x20                ; initial code starts at offset 0x20

;--    SECTIONrom_code
```

```
        .global ___reset
___reset:
rom_start:
```



```
    move.b  #$9,PGSEL           ; config PG0/DTACK to GPI/O,input
    move.w  #$2410,PLLCR        ; sysclk = VCO/1, enable clko
    move.l  #MON_STACKTOP,A7    ; Install stack pointer
    move.w  #$2700,sr           ; mask off all interrupts
    move.b  #$1C,SCR            ; enable bus error timeout bit
    move.w  #0,RTCWD            ; disable watch dog
    move.b  #$03,PFSEL          ; select A23-A20, CLKO, CSA1
    move.b  #$00,PBSEL          ; Config port B for chip select A,B,C and D
    move.b  #$00,PESEL          ; select *DWE
    move.w  #$08,ICEMCR         ; disable ICEM vector hardmap
    move.w  #$07,ICEMSR         ; clear level 7 interrupt

;*****************************
; Init Code for 512Kx16 Flash
;*****************************
    move.w  #$0400,GRPBASEA     ; GROUPA BASE(FLASH), Start address=0x800000
    move.w  #$0189,CSA          ; 2MB each, 0ws, FLASH,

;*********************************
; Init Code 4Mx16 (12/10) EDO DRAM
;*********************************
    move.w  #$8F00,DRAMCFG      ; DRAM Memory Config Register
                              ; Col Address = PA1-PA10; Row Address = PA11-PA22
    move.w  #$9667,DRAMCTL      ; DRAM Control Register, 0ws
    move.w  #$0000,GRPBASED     ; GROUPA BASE(DRAM),
                                ;Start address=0x000000-0x7FFFFF
    move.w  #$069F,CSD          ; enable DRAM cs

;*********************************
; Init Code for EMUCS
;*********************************
    move.w  #$30,EMUCS          ; 3ws

    clr.l       d0
    clr.l       d1
    clr.l       d2
    clr.l       d3
    clr.l       d4
    clr.l       d5
    clr.l       d6
    clr.l       d7

;*****************************************************************************
;
;   LCD Temp init for screen protection
;
;*****************************************************************************

    move.b   #0,PCSEL
    move.b   #0,PCPDEN
```

```
    move.l   #$4000,LSSA
    move.w   #160,LXMAX
    move.w   #239,LYMAX
    move.b   #10,LVPW
    move.b   #$08,LPICF
    move.b   #$01,LPOLCF
    move.b   #$00,LACDRC
    move.b   #$02,LPXCD
    move.b   #$14,LRRA
    move.b   #$00,LPOSR
    move.b   #$00,LCKCON          ; disable LCDC
    move.b   #$80,LCKCON          ; enable LCDC, 0ws, 16-bit


;*****************************************************************
;  Interrupt controller
;*****************************************************************

       move.b #$40,IVR
       move.l #$007fffff,IMR; mask all interrupts except *EMUIRQ,
                                          ; since it is used for the abort
       button
                                          ; on this board

       .extern__start
       JMP __start  ; jump to MW startup code
```


D.2 INITIALIZATION CODE OF SDS MONITOR (MONITOR.H)

```
        .option list="off"
*******************************************************************************
*       EZ328EMU.H - Configuration file for Motorola's M68EZ328ADS
*******************************************************************************

*******************************************************************************
*       CPU OPTIONS
*******************************************************************************

        opt     p=68000

M328BASE  equ $FFFFF000

; SIM28 System Configuration Registers
SCR             equ     (M328BASE+$000)

; Chip Select Registers
GRPBASEA        equ     (M328BASE+$100)
GRPBASEB        equ     (M328BASE+$102)
GRPBASEC        equ     (M328BASE+$104)
GRPBASED        equ     (M328BASE+$106)
CSA             equ     (M328BASE+$110)
CSB             equ     (M328BASE+$112)
CSC             equ     (M328BASE+$114)
CSD             equ     (M328BASE+$116)
DRAMCFG         equ     (M328BASE+$C00)
```

```
DRAMCTL         equ     (M328BASE+$C02)
EMUCS           equ     (M328BASE+$118)
CSCTR           equ     (M328BASE+$150)
```



```
; PLL Registers
PLLCR           equ     (M328BASE+$200) ; Control Reg
PLLFSR          equ     (M328BASE+$202) ; Freq Select Reg
PLLTSR          equ     (M328BASE+$204) ; Test Reg

; Power Control Registers
PCTLR           equ     (M328BASE+$206) ; Control Reg

; Interrupt Registers
IVR             equ     (M328BASE+$300) ; Interrupt Vector Reg
ICR             equ     (M328BASE+$302) ; Interrupt Control Reg
IMR             equ     (M328BASE+$304) ; Interrupt Mask Reg
ISR             equ     (M328BASE+$30C) ; Interrupt Status Reg
IPR             equ     (M328BASE+$310) ; Interrupt Pending Reg

; PIO Registers
                        ; Port A Registers
PADIR           equ     (M328BASE+$400) ; Direction Reg
PADATA          equ     (M328BASE+$401) ; Data Reg
PAPUEN          equ     (M328BASE+$402) ; Pullup Enable Reg
                        ; Port B Registers
PBDIR           equ     (M328BASE+$408) ; Direction Reg
PBDATA          equ     (M328BASE+$409) ; Data Reg
PBPUEN          equ     (M328BASE+$40A) ; Pullup Enable Reg
PBSEL           equ     (M328BASE+$40B) ; Select Reg
                        ; Port C Registers
PCDIR           equ     (M328BASE+$410) ; Direction Reg
PCDATA          equ     (M328BASE+$411) ; Data Reg
PCPDEN          equ     (M328BASE+$412) ; Pull-down Enable Reg
PCSEL           equ     (M328BASE+$413) ; Select Reg
                        ; Port D Registers
PDDIR           equ     (M328BASE+$418) ; Direction Reg
PDDATA          equ     (M328BASE+$419) ; Data Reg
PDPUEN          equ     (M328BASE+$41A) ; Pullup Enable Reg
PDSEL           equ     (M328BASE+$41B) ; port D select
PDPOL           equ     (M328BASE+$41C) ; Polarity Reg
PDIRQEN         equ     (M328BASE+$41D) ; IRQ Enable Reg
PDIRQEDGE       equ     (M328BASE+$41F) ; IRQ Edge Reg
                        ; Port E Registers
PEDIR           equ     (M328BASE+$420) ; Direction Reg
PEDATA          equ     (M328BASE+$421) ; Data Reg
PEPUEN          equ     (M328BASE+$422) ; Pullup Enable Reg
PESEL           equ     (M328BASE+$423) ; Select Reg
                        ; Port F Registers
PFDIR           equ     (M328BASE+$428) ; Direction Reg
PFDATA          equ     (M328BASE+$429) ; Data Reg
PFPUEN          equ     (M328BASE+$42A) ; Pullup Enable Reg
PFSEL           equ     (M328BASE+$42B) ; Select Reg
                        ; Port G Registers
PGDIR           equ     (M328BASE+$430) ; Direction Reg
PGDATA          equ     (M328BASE+$431) ; Data Reg
```

```
PGPUEN          equ     (M328BASE+$432) ; Pullup Enable Reg
PGSEL           equ     (M328BASE+$433) ; Select Reg

; PWM Registers
PWMC            equ     (M328BASE+$500) ; Control Reg
PWMS            equ     (M328BASE+$502) ; Sample Reg
PWMCNT          equ     (M328BASE+$504) ; Count Reg

; Timer Registers
                          ; Timer 1 Registers
TCTL            equ     (M328BASE+$600) ; Control Reg
TPRER           equ     (M328BASE+$602) ; Prescalar Reg
TCMP            equ     (M328BASE+$604) ; Compare Reg
TCR             equ     (M328BASE+$606) ; Capture Reg
TCN             equ     (M328BASE+$608) ; Counter
TSTAT           equ     (M328BASE+$60A) ; Status Reg

; SPI Registers
SPIMDATA        equ     (M328BASE+$800) ; Control/Status Reg
SPIMCONT        equ     (M328BASE+$802) ; Data Reg

; UART Registers
USTCNT          equ     (M328BASE+$900) ; Status Control Reg
UBAUD           equ     (M328BASE+$902) ; Baud Control Reg
UARTRX          equ     (M328BASE+$904) ; Rx Reg
UARTTX          equ     (M328BASE+$906) ; Tx Reg
UARTMISC        equ     (M328BASE+$908) ; Misc Reg
UARTNIPR        equ     (M328BASE+$90A) ; None-Integer Prscaler reg

; LCDC Registers
LSSA            equ     (M328BASE+$A00) ; Screen Start Addr Reg
LVPW            equ     (M328BASE+$A05) ; Virtual Page Width Reg
LXMAX           equ     (M328BASE+$A08) ; Screen Width Reg
LYMAX           equ     (M328BASE+$A0A) ; Screen Height Reg
LCXP            equ     (M328BASE+$A18) ; Cursor X Position
LCYP            equ     (M328BASE+$A1A) ; Cursor Y Position
LCWCH           equ     (M328BASE+$A1C) ; Cursor Width & Height Reg
LBLKC           equ     (M328BASE+$A1F) ; Blink Control Reg
LPICF           equ     (M328BASE+$A20) ; Panel Interface Config Reg
LPOLCF          equ     (M328BASE+$A21) ; Polarity Config Reg
LACDRC          equ     (M328BASE+$A23) ; ACD (M) Rate Control Reg
LPXCD           equ     (M328BASE+$A25) ; Pixel Clock Divider Reg
LCKCON          equ     (M328BASE+$A27) ; Clocking Control Reg
LRRA            equ     (M328BASE+$A29) ; Refresh Rate Adjust reg
LPOSR           equ     (M328BASE+$A2D) ; Panning Offset Reg
LFRCM           equ     (M328BASE+$A31) ; Frame Rate Control Mod Reg
LGPMR           equ     (M328BASE+$A33) ; Gray Palette Mapping Reg
LPWM            equ     (M328BASE+$A36) ; contrast control Reg

; RTC Registers
RTCHMSR         equ     (M328BASE+$B00) ; Hrs Mins Secs Reg
RTCALM0R        equ     (M328BASE+$B04) ; Alarm Register
RTCDAY          equ     (M328BASE+$B08) ; RTC date reg
RTCWD           equ     (M328BASE+$B0A) ; RTC watch dog timer reg
RTCCTL          equ     (M328BASE+$B0C) ; Control Reg
```

```
RTCISR          equ     (M328BASE+$B0E) ; Interrupt Status Reg
RTCIENR         equ     (M328BASE+$B10) ; Interrupt Enable Reg
RSTPWCH         equ     (M328BASE+$B12) ; Stopwatch Minutes
```



```
;ICEM registers
ICEMACR         equ     (M328BASE+$D00)
ICEMAMR         equ     (M328BASE+$D04)
ICEMCCR         equ     (M328BASE+$D08)
ICEMCMR         equ     (M328BASE+$D0A)
ICEMCR          equ     (M328BASE+$D0C)
ICEMSR          equ     (M328BASE+$D0E)

*****************************************************************************
*       ADDRESSING OPTIONS
*****************************************************************************
;INTERVAL:       equ     2
INTERVAL:       equ     1
*****************************************************************************
*       I/O DEVICE OPTIONS
*****************************************************************************
*****************************************************************
*   WARNING: This file redefines _usr_gchar bypassing the BSP   *
*****************************************************************

***************************************
* Setting for ST16C2552
***************************************
#define DEV_IN  ST16C2552
#define DEV_OUT ST16C2552
BASE_IN:        equ     $FFFD0000       ; base address
BASE_OUT:       equ     $FFFD0000       ; base address
BAUD_IN:        equ     576             ; baud rates = BAUD_IN/100 bps
BAUD_OUT:       equ     576             ; baud rates = BAUD_OUT/100 bps

***************************************
* Setting for EZ-UART
***************************************
;#define DEV_IN  EZUART
;#define DEV_OUT EZUART
;BASE_IN:        equ     $fffd8000       ; base address not applicable
;BASE_OUT:       equ     $fffd8000       ; base address not applicable
;BAUD_IN:        equ     1152            ; baud rates = 115200 bps
;BAUD_OUT:       equ     1152            ; baud rates = 115200 bps

***************************************
* Seting for ADI PORT on EZ-ADS
***************************************
;#define DEV_IN  ADIPORT
;#define DEV_OUT ADIPORT
;BASE_IN:        equ     $fffd8000       ; base address of input device
;BASE_OUT:       equ     $fffd8000       ; base address of output device
;BAUD_IN:        equ     0               ; baud rates not applicable
;BAUD_OUT:       equ     0               ; baud rates not applicable

***************************************
```

```
* Seting for MC68681 on EZ-ADS
*        Supported baudrate: 19200
*                            9600
************************************
;#define DEV_IN         MC68681
;#define DEV_OUT        MC68681
;#define PNAME_IN  A
;#define PNAME_OUT A
;BASE_IN:        equ     $fffd0001       ; base address of input device
;BASE_OUT:       equ     $fffd0001       ; base address of output device
;BAUD_IN:        equ     19200           ; baud rates (bits/sec)
;BAUD_OUT:       equ     19200           ; baud rates (bits/sec)
;ACR_VAL:        equ     $80             ; choose baud set 2 (bit#7=1)
;IRQ_MASK:       equ     $00
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

************************************************
* Seting for MC68681 on EZ-ADS with BaudRate=57600bps
************************************************
;#define DEV_IN         MC68681
;#define DEV_OUT        MC68681
;#define PNAME_IN  A
;#define PNAME_OUT A
;BASE_IN:        equ     $fffd0001       ; base address of input device
;BASE_OUT:       equ     $fffd0001       ; base address of output device
;BAUD_IN:        equ     0
;BAUD_OUT:       equ     0
;ACR_VAL:        equ     $60
;IRQ_MASK:       equ     $00
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

*****************************************************************************
*       INTERRUPT BEHAVIOR OPTIONS
*****************************************************************************
#define STOP_LEVEL      7

#define VECT_STOP       $47             ; ABORT interrupt

        ; ************************************************
        ; If your board hardware requires you to clear the
        ; "read" interrupt, then you must include the necessary
        ; code inside the INT_RCLEAR macro below.
        ; ************************************************
INT_RCLEAR:     macro
        endm

        ; ************************************************
        ;  "stop" interrupt
        ;  We will only stop the target if the ABORT switch is pressed. We
        ;  will not stop for HOST_NMI or EXT_NMI interrupts.
        ;  The Non-Maskable Interrupt status is in the NMI_STATUS register
        ;  which is in even-byte locations at NMI_STATUS. Bit #0 of the
        ; first byte at NMI_STATUS indicates that the ABORT switch was pressed.
        ; ************************************************
INT_STOP:       macro
```

```
        move.w  #$7,ICEMSR      ; clear level 7 interrupt
        or.w    #$FF80,ISR
        endm                    ; return non-zero in D0 if ABORT
```



```
INT_DOG:        macro
        endm


       ; ************************************************
       ;  Macros for use at application startup (EXEC)
       ;  and stop (DONE). Normally these are empty.
       ; ************************************************
USR_EXEC:       macro
        endm

USR_DONE:       macro
        endm


*************************************************************************
*       RESET OPTIONS
*************************************************************************

       ; ************************************************
       ;  Contents of the "usr_reset" section. In the usual
       ;  case, this is two longwords long, with the first word
       ;  containing the value with which the processor
       ;  loads up its stack pointer, and the second of
       ;  which is the value with which the processor
       ;  loads up its program counter. This will bring up
       ;  the monitor on a hardware reset.
       ; ************************************************
USR_RESET:      macro
        dc.l    MON_STACKTOP            ; stack pointer (default)
        dc.l    reset$$$                ; program counter (default)
        dcb.l   6,0
        endm

       ; ************************************************
       ;  Hard reset macro - contains very high priority
       ;  instructions which must be executed immediately
       ;  upon a hardware reset.
       ; ************************************************

RESET_HARD:    macro
    move.b  #$9,PGSEL           ; config PG0/DTACK to GPI/O,input
    move.w  #$2410,PLLCR        ; sysclk = VCO/1, enable clko
    move.l  #MON_STACKTOP,A7    ; Install stack pointer
    move.w  #$2700,sr           ; mask off all interrupts
    move.b  #$1C,SCR            ; enable bus error timeout bit
    move.w  #0,RTCWD            ; disable watch dog
    move.b  #$03,PFSEL          ; select A23-A20, CLKO, CSA1
    move.b  #$00,PBSEL          ; Config port B for chip select A,B,C and D
    move.b  #$00,PESEL          ; select *DWE
    move.w  #$08,ICEMCR         ; disable ICEM vector hardmap
```

```
    move.w  #$07,ICEMSR         ; clear level 7 interrupt

;*****************************
; Init Code for 512Kx16 Flash
;*****************************
    move.w  #$0400,GRPBASEA     ; GROUPA BASE(FLASH), Start address=0x800000
    move.w  #$0189,CSA          ; 2MB each, 0ws, FLASH,

;*********************************
; Init Code 4Mx16 (12/10) EDO DRAM
;*********************************
    move.w  #$8F00,DRAMCFG      ; DRAM Memory Config Register
                              ; Col Address = PA1-PA10; Row Address = PA11-PA22
    move.w  #$9667,DRAMCTL      ; DRAM Control Register, 0ws
    move.w  #$0000,GRPBASED     ; GROUPA BASE(DRAM), Start address=0x000000-
       0x7FFFFF
    move.w  #$069F,CSD          ; enable DRAM cs

;*********************************
; Init Code for EMUCS
;*********************************
    move.w  #$30,EMUCS          ; 3ws

    clr.l       d0
    clr.l       d1
    clr.l       d2
    clr.l       d3
    clr.l       d4
    clr.l       d5
    clr.l       d6
    clr.l       d7

;*************************************************************************
;
;   LCD Temp init for screen protection
;
;*************************************************************************

    move.b   #0,PCSEL
    move.b   #0,PCPDEN

    move.l   #$4000,LSSA
    move.w   #160,LXMAX
    move.w   #239,LYMAX
    move.b   #10,LVPW
    move.b   #$08,LPICF
    move.b   #$01,LPOLCF
    move.b   #$00,LACDRC
    move.b   #$02,LPXCD
    move.b   #$14,LRRA
    move.b   #$00,LPOSR
    move.b   #$00,LCKCON            ; disable LCDC
    move.b   #$80,LCKCON            ; enable LCDC, 0ws, 16-bit

;*************************************************************************
```

```
;
; Initialize I/O to avoid contention
;
;*********************************************************************
```



```
;       TBD since the contention does not affect ADS operation


;*********************************************************************
;
; Program Interrupt Controller
;
;*********************************************************************

        move.b #$40,IVR
        move.l #$007FFFFF,IMR           ;enable NMI interrupt

        endm

        ; ************************************************
        ;  Soft reset macro - contains instructions
        ;  which must be executed upon a processor reset.
        ; ************************************************
RESET_SOFT:     macro

                ; ***************************************
                ; * Setting for MC68681
                ; ***************************************
                .if "\{DEV_IN}"=="MC68681"
MC68681ctur     equ  (BASE_IN+OFF6)
MC68681ctlr     equ  (BASE_IN+OFF7)
                move.b #IRQ_MASK,MC68681imr  ;set value of IMR
                move.b #ACR_VAL,MC68681acr   ;set value of ACR
                ; Setting for baudrates = 57600bps
                move.b #00,MC68681ctur
                move.b #02,MC68681ctlr ; divider=2
                .endif

        endm

*************************************************************************
*       OTHER DEFINITIONS
*************************************************************************

USER_DEFS:      macro

                ; ***************************************
                ; * Setting for MC68681
                ; ***************************************
                .if "\{DEV_IN}"=="MC68681"
                xdef  MC68681acr
                xdef  MC68681imr

                SECTION     usr_ram
                MC68681acr: ds.b 1 ;ACR register
```

```
MC68681imr: ds.b 1 ;IMR register
.endif

endm
```

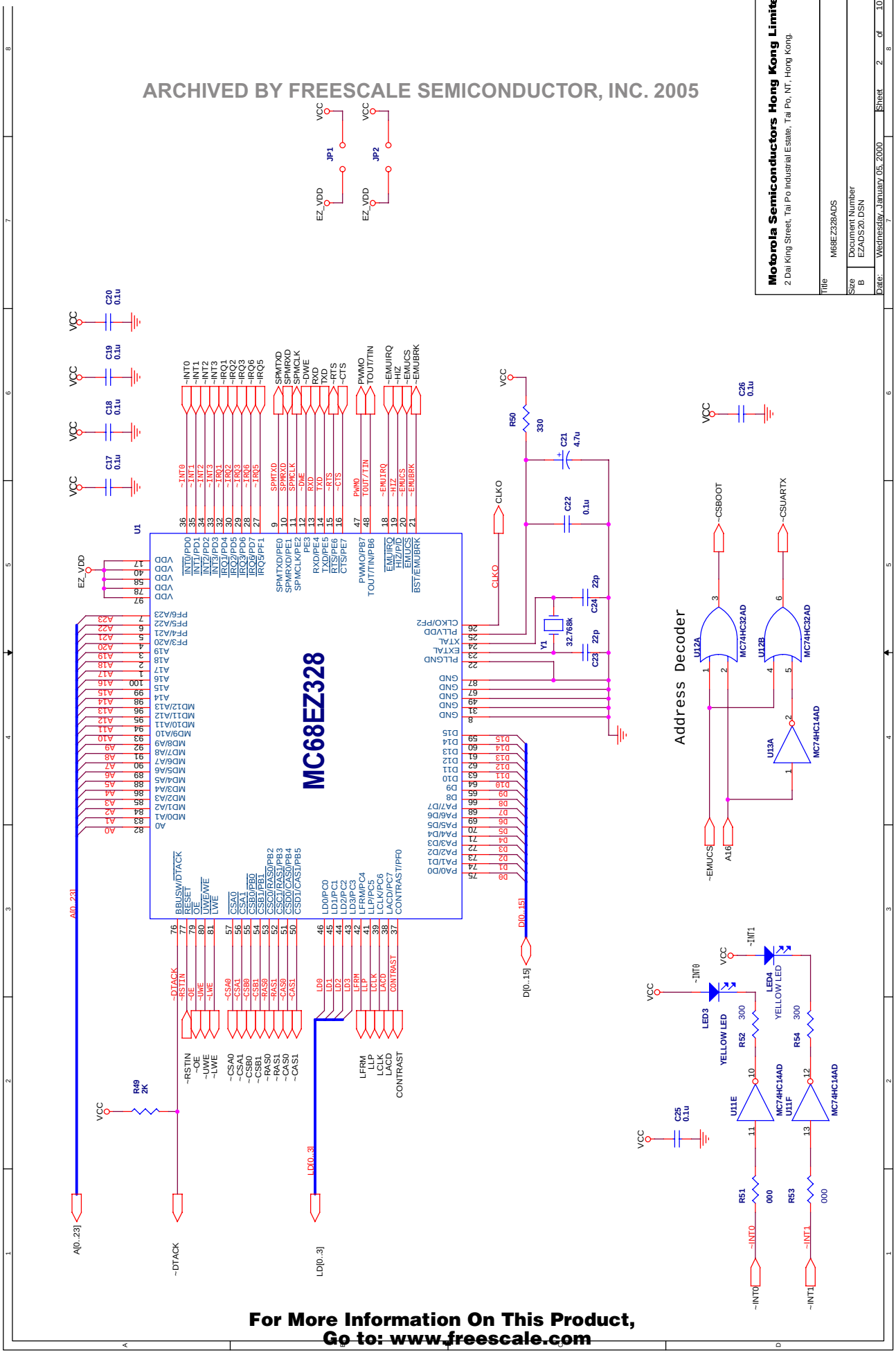
APPENDIX E
SCHEMATICS

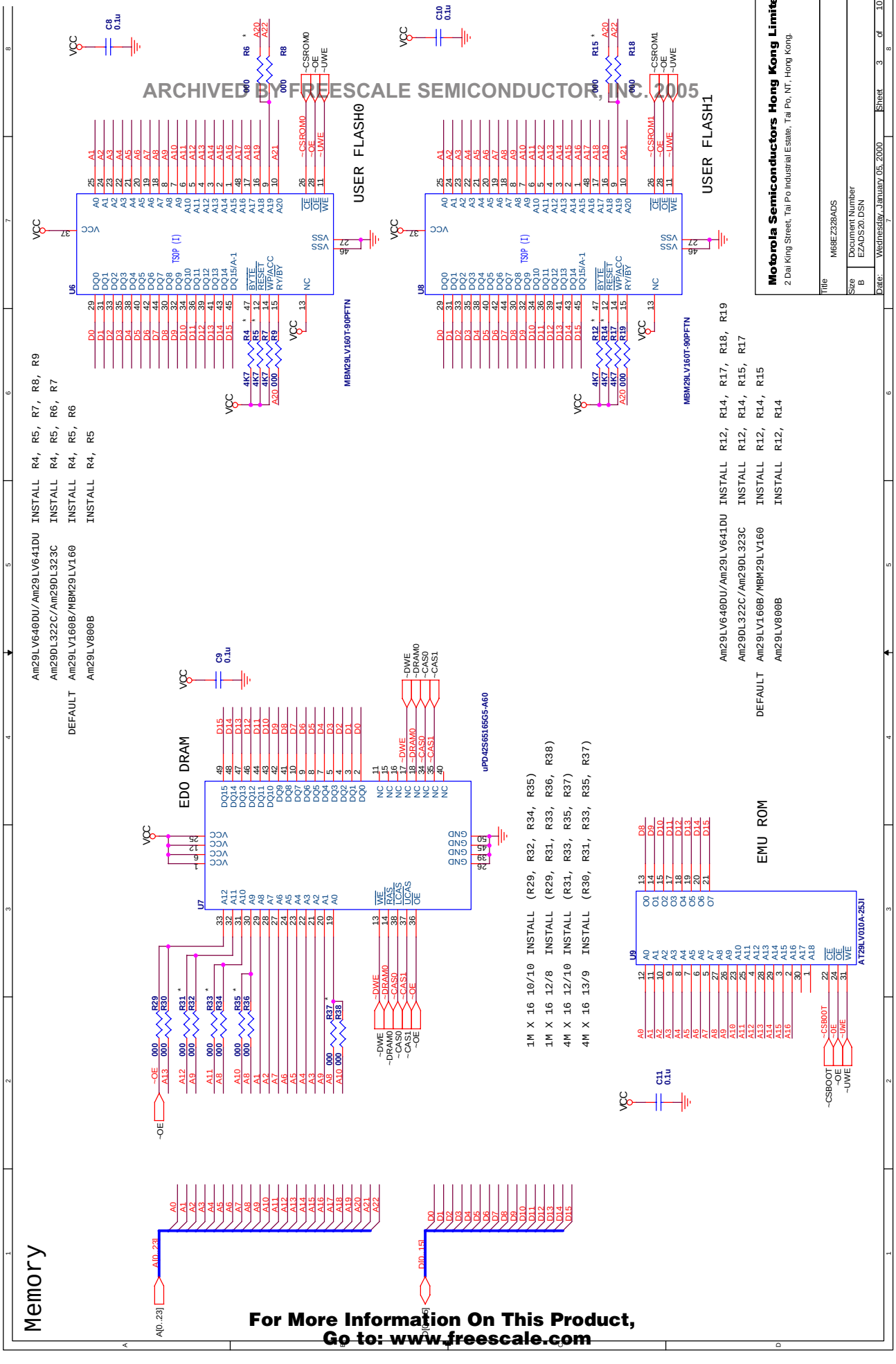
E.1 M68EZ328ADS V2.0

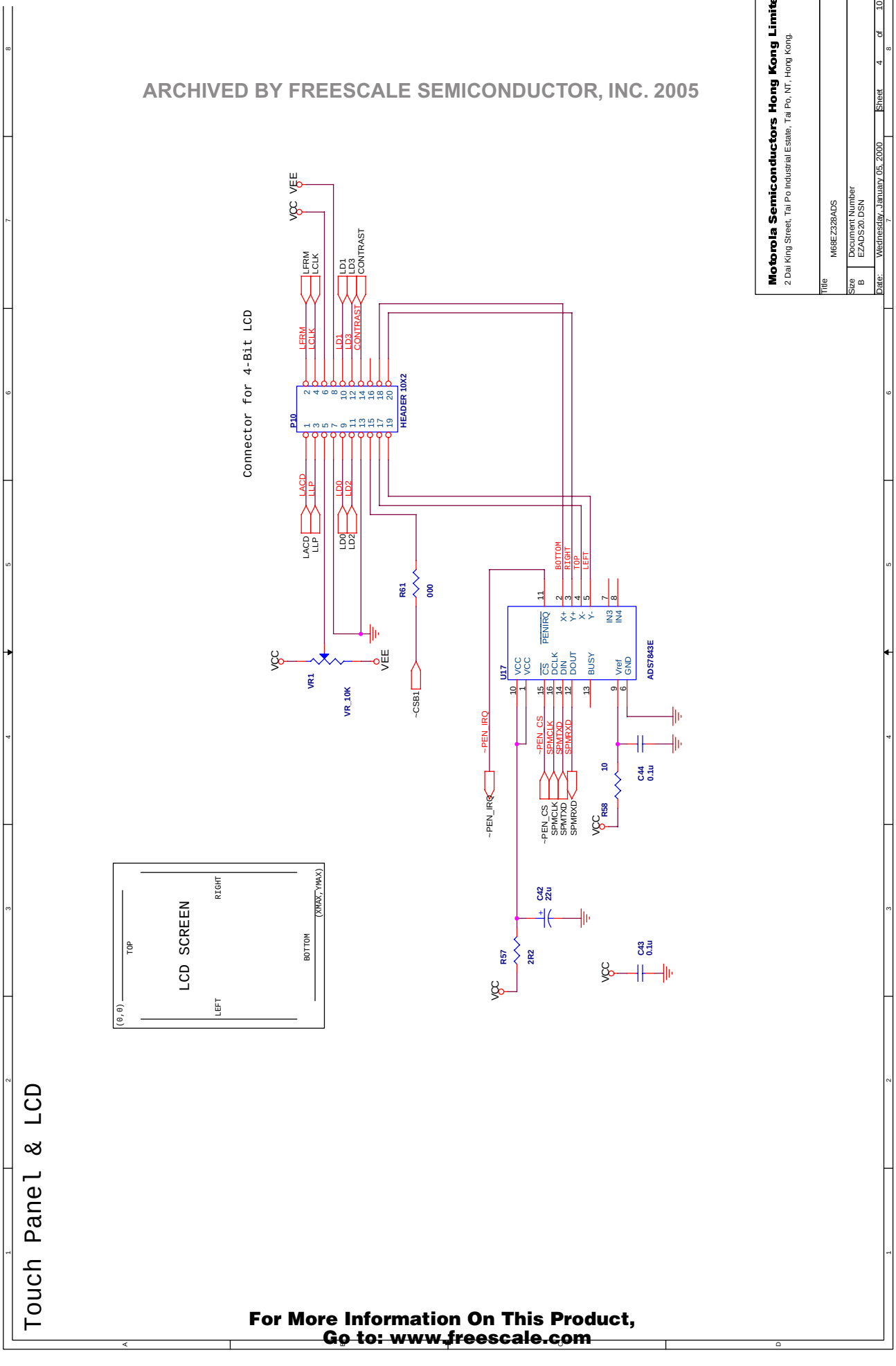
Motorola Semiconductors Hong Kong Ltd.
2 Dai King St, Tai Po Ind Est, Tai Po, NT, Hong Kong.

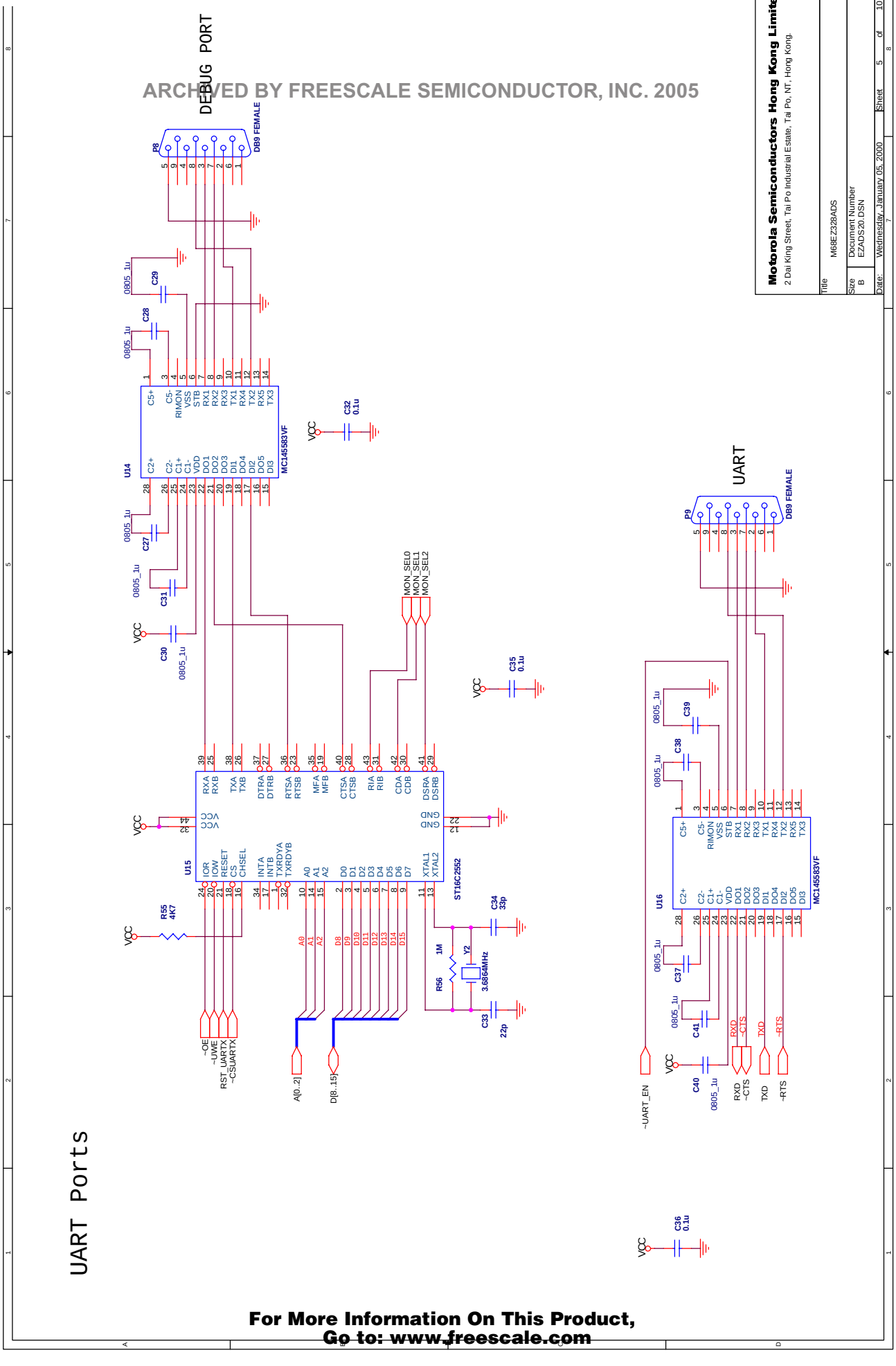
Schematic: M68EZ328ADS
Version: 2.0

File Name: EZADS20.DSN
Last Updated: Teddy Wong on 5 Nov, 1999



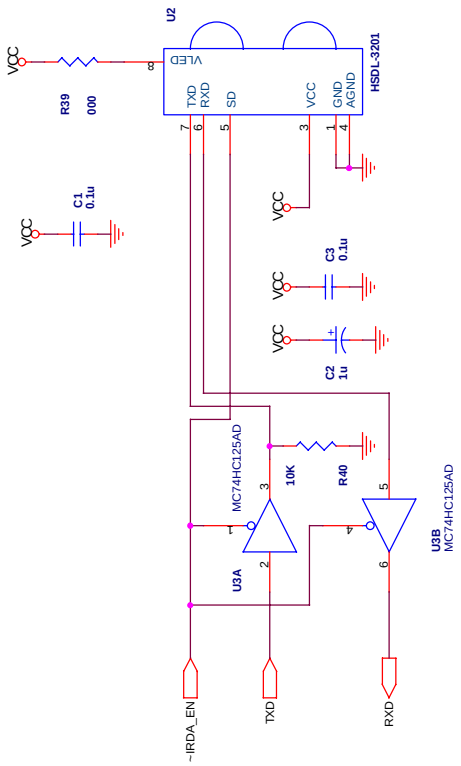




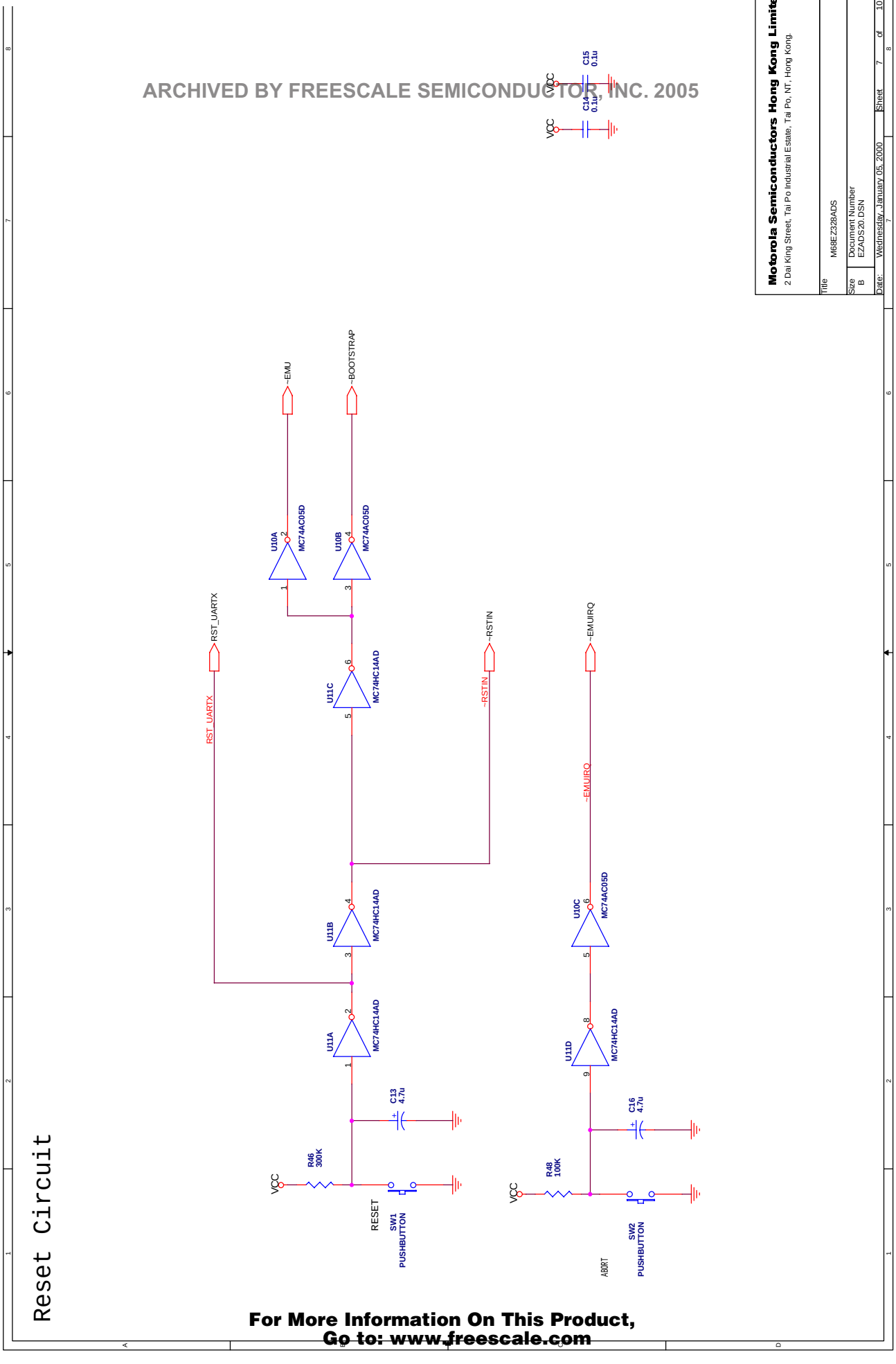


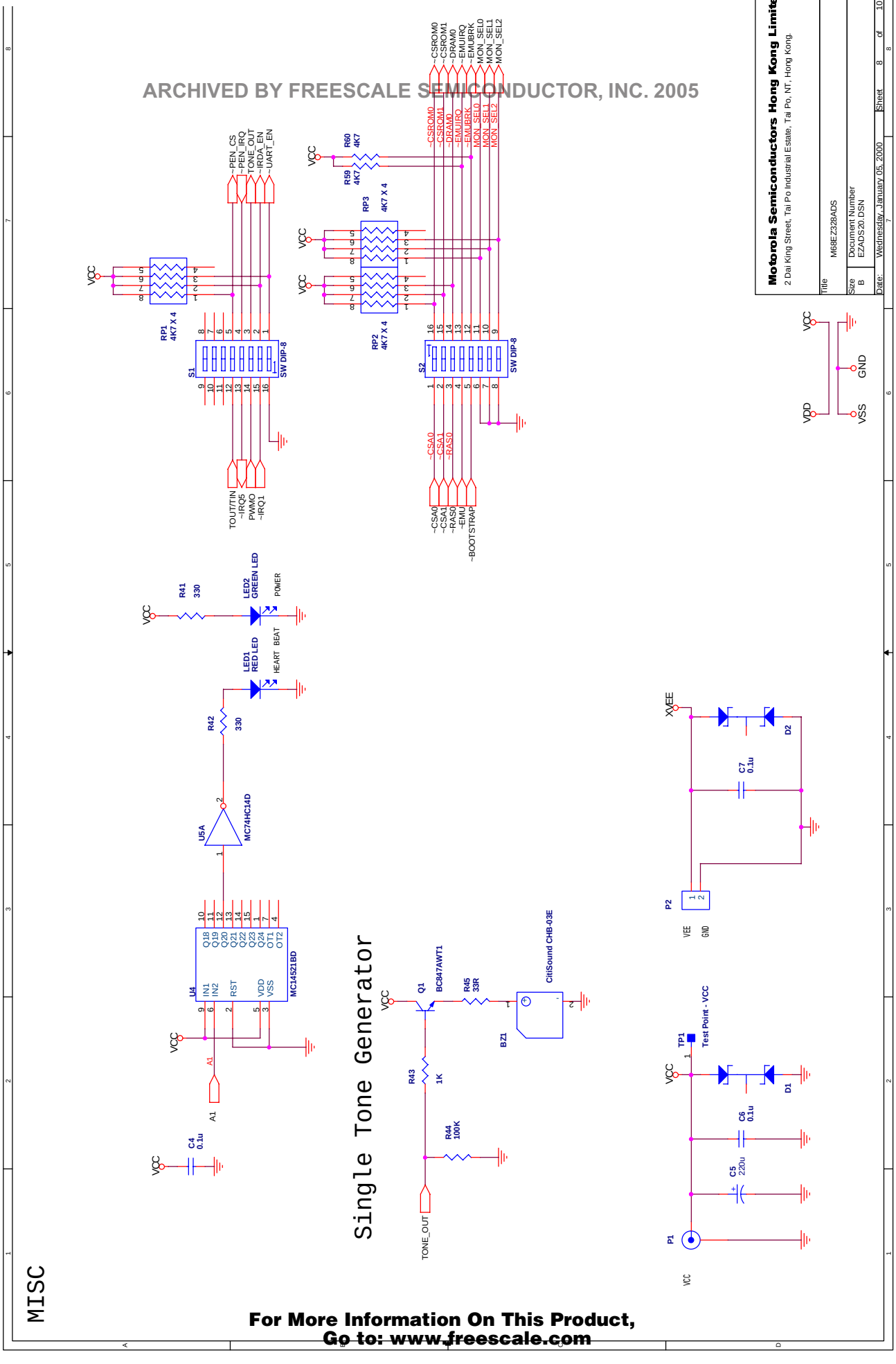
Motorola Semiconductors Hong Kong Limited
2 Dai King Street, Tai Po Industrial Estate, Tai Po, NT, Hong Kong.

Title	M68EZ328ADS		
Size B	Document Number EZADS20.DSN		Rev 2.0
Date: Wednesday, January 05, 2000		Sheet 6	of 10

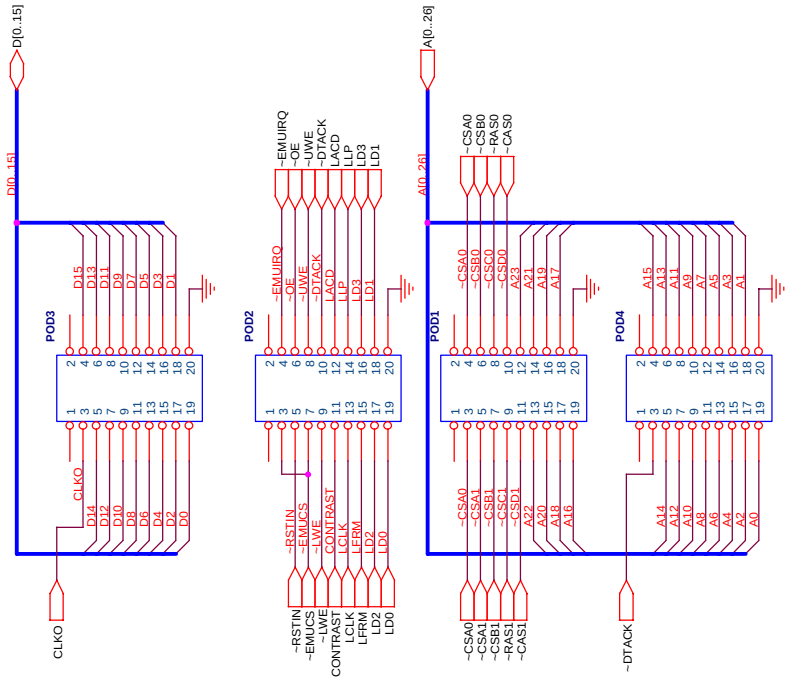


IrDA





Logic Analyzer Connectors



Motorola Semiconductors Hong Kong Limited
2 Dai King Street, Tai Po Industrial Estate, Tai Po, NT, Hong Kong.

Title	M68EZ328ADS	
Size B	Document Number EZADS20.DSN	Rev 2.0
Date: Wednesday, January 05, 2000	Sheet 9 of 10