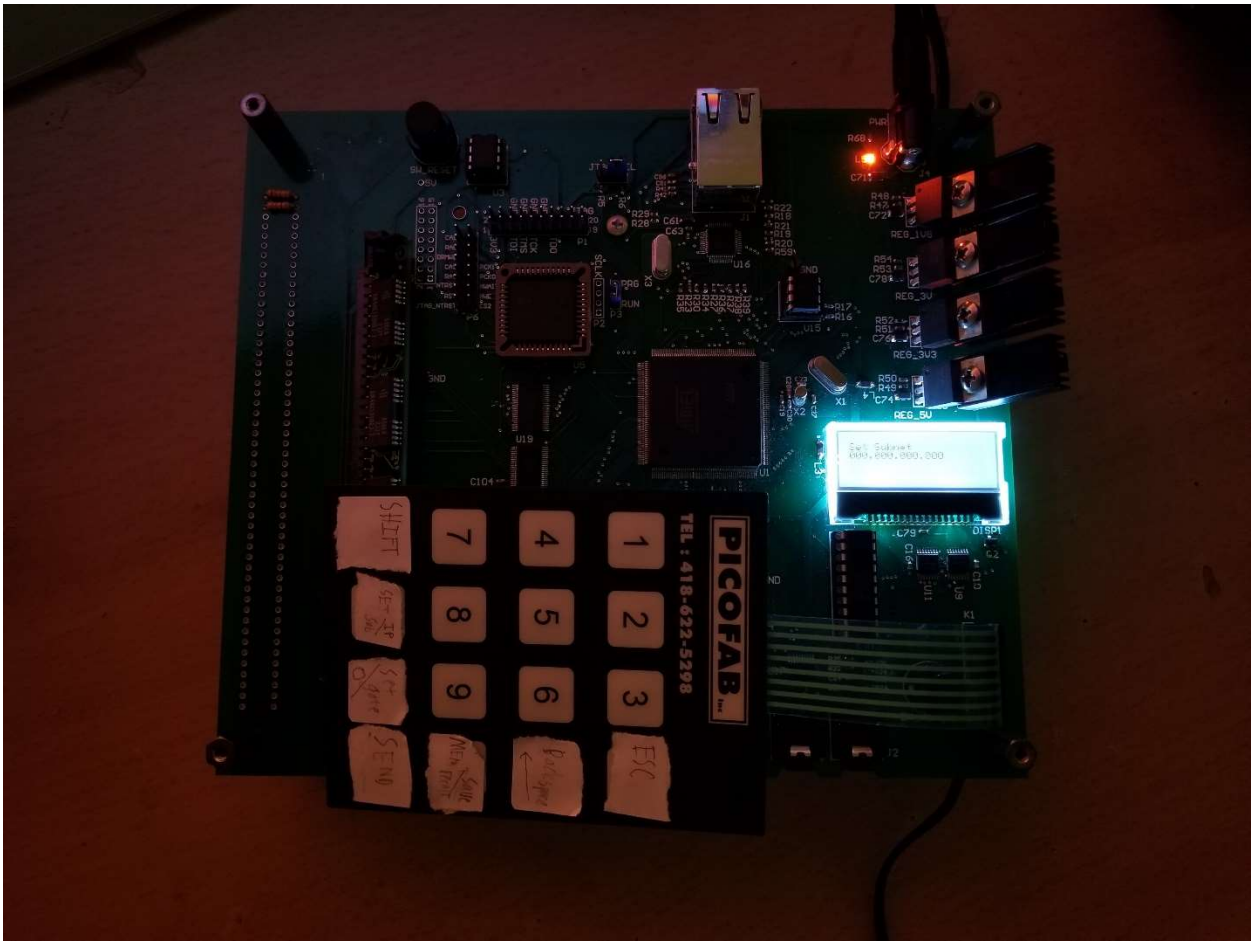
VoIP phone User Manual



The EE52 VoIP phone is a phone that can communicate with other EE52 VoIP phones. The user controls the phone by using a 128x32 display and a 4x4 keypad. There is a headphone and microphone jack that a headset can be plugged into, and there is a button for putting the phone “on-the-hook” to hangup the call.

Will Werst

Contents

System Specifications

Processor	Atmel AT91RM9200
Input power	8V-12V DC 5.5mm barrel jack
Display	128x32
Headphone Jack	3.5mm mono audio jack
Microphone Jack	3.5mm
Keypad	4x4 button keypad

Startup

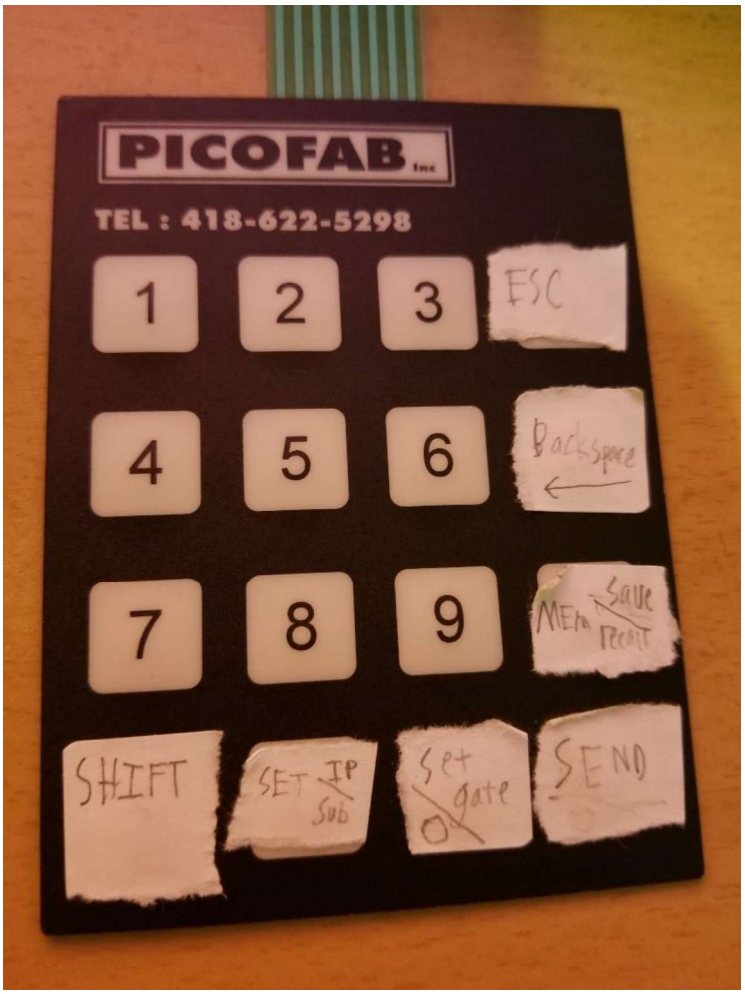
Plug the system into the wall with the provided 12V DC wall power supply. The system will initialize and run a full diagnostic test on the onboard memory. After the SRAM and DRAM have been thoroughly tested for errors, the display backlight will blink and then the status “Idle” will appear on the screen. The phone is now fully booted. Proceed to the next section “Interfacing with the phone”

Resetting the system

Should something go awry in the system, and a reset is needed, simply press the reset button located along the same side of the pcb as the ethernet jack. The system will reset, the display will blink once the memory check completes, and the system is then ready to go.

Interfacing with the phone

Keypad



The keypad is a 4x4 keypad. Some keys are multiplexed using a shift key at the bottom left of the keyboard. The shift key toggles the keyboard between shift mode and normal mode. Note, this is different than the way most keyboards operate due to a limitation in the ability to detect multiple key presses.

Pressing Set IP/Sub allows the user to set the IP address and subnet mask using the numerical keys.

Pressing Set gate allows the user to set the gateway IP address.

Pressing Mem Save/Recall allows the user to save the current address configuration to an address location, or recall the IP settings from a memory location.

Pressing Send should initiate a call.

Note, due to a bug in the LWIP library, it is not possible to set more than 3 address settings, otherwise the system will crash.

Display

The display is a 128x32 display, divided into 4 rows. Only the top two rows are used.

The top row displays the current menu the system is in, or the status of the system if a call is in progress.

State	1st Row message
Idle	Idle
Phone off hook	Off Hook
Ringing remote user	Ringing
Connecting to remote user	Connecting
Connected to remote user	Connected
Setting IP address	Set IP
Setting subnet mask	Set Subnet
Setting gateway IP address	Set Gateway
Saving settings to memory	Memory Save
Recalling settings from memory	Memory Recall
Settings recalled from memory	Recalled Message
Error occurred in software	Illegal Message

The 2nd row displays auxiliary information for a menu, such as the IP address being set, the subnet mask, or the memory location for recall. The keyboard can be used to input data here, and save with the Send button.

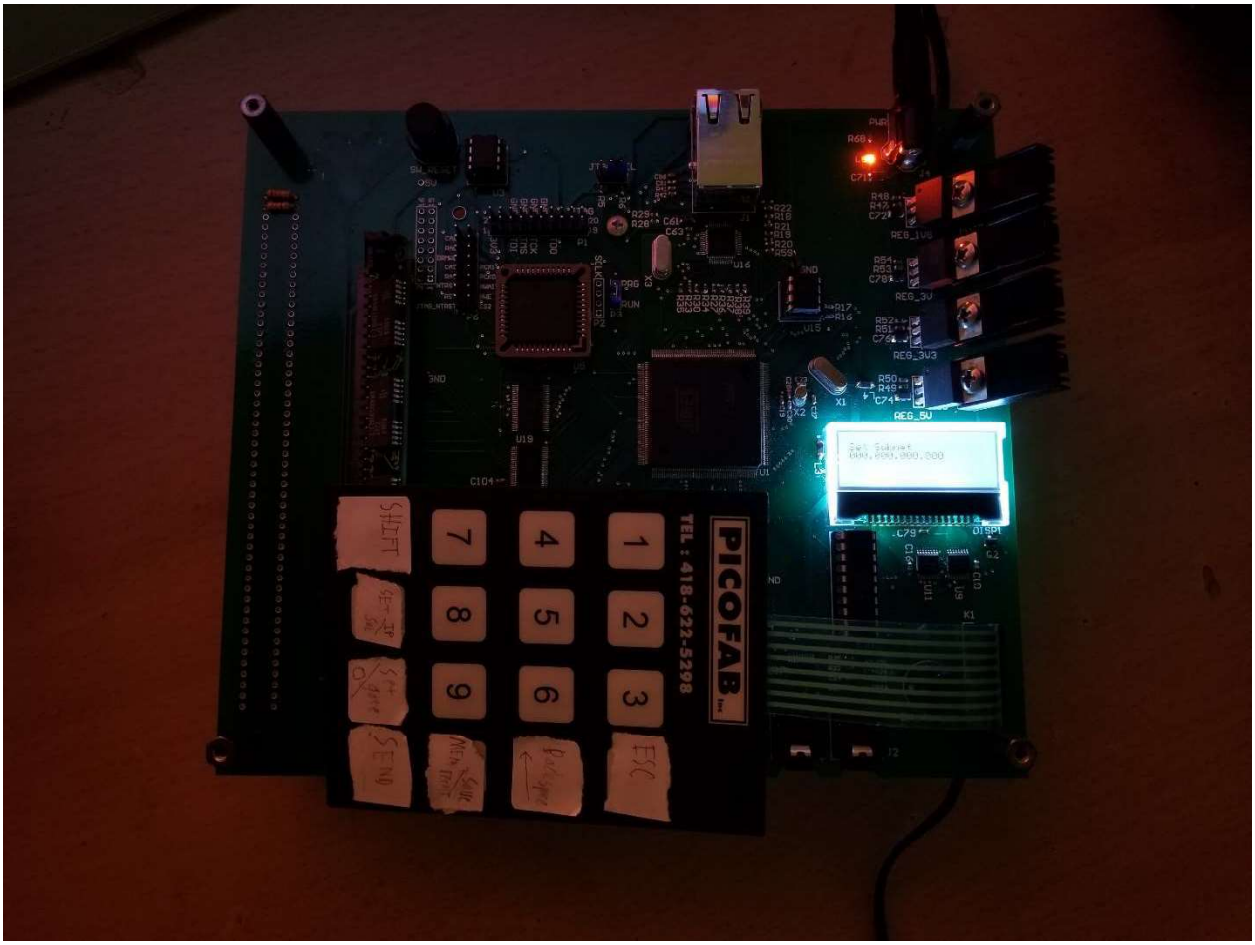
Headset

Two 3.5mm audio jacks are mounted next to the keyboard. J2 is the microphone, and J3 is the headset connection. The software has volume set capability, but the code must be recompiled to change it, so it is recommended the user use an adjustable volume headset.

Ethernet

Ethernet is located on the same side of PCB as the power jack. The ethernet is designed to work in 10Mbps mode with RJ-45 connector-based cables. Currently, ethernet is not supported by this phone (what a great phone, I know).

VoIP phone Technical Documentation

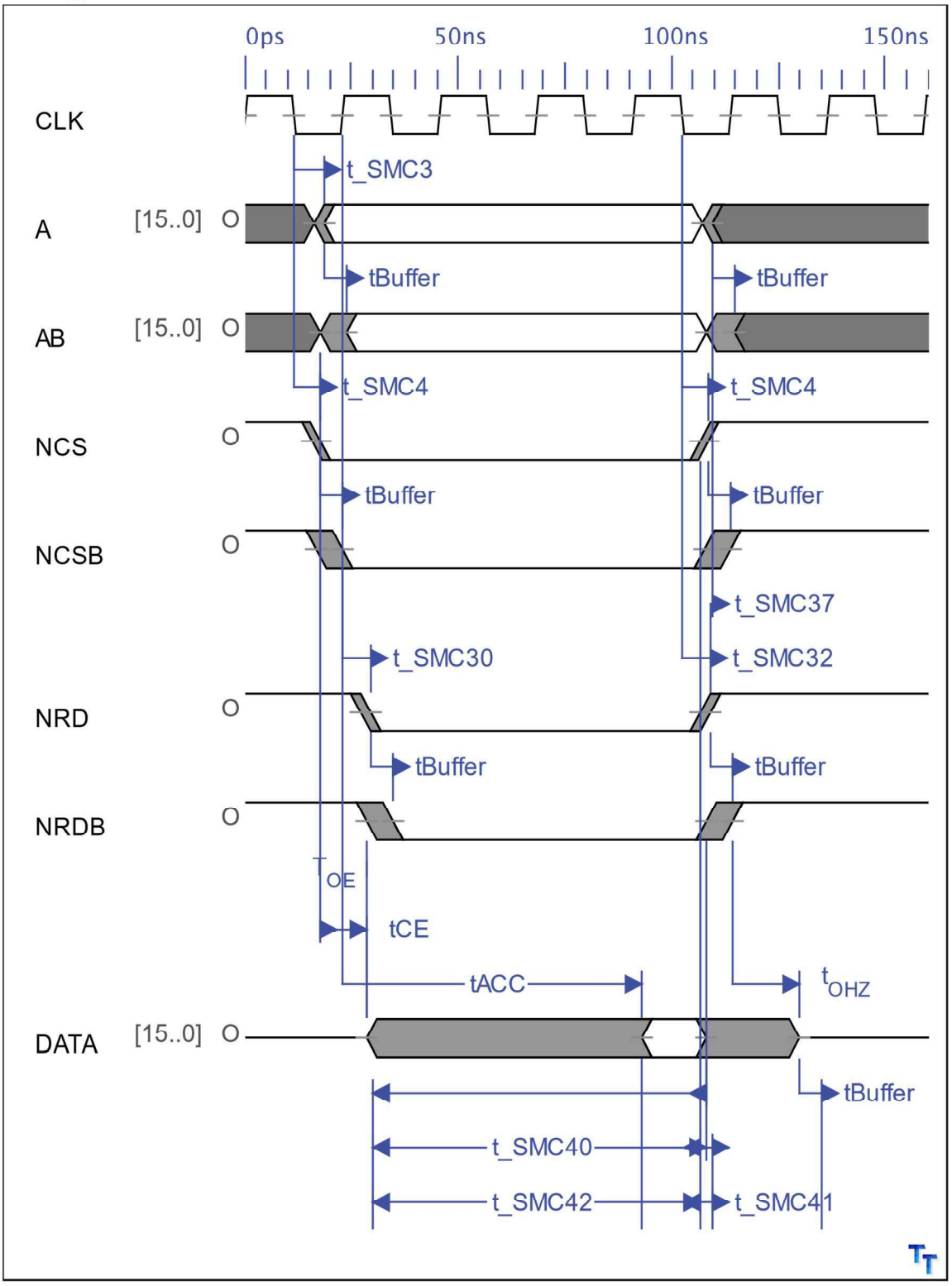


The EE52 VoIP phone is a phone that can communicate with other EE52 VoIP phones. The user controls the phone by using a 128x32 display and a 4x4 keypad. There is a headphone and microphone jack that a headset can be plugged into, and there is a button for putting the phone “on-the-hook” to hangup the call.

Will Werst

Contents

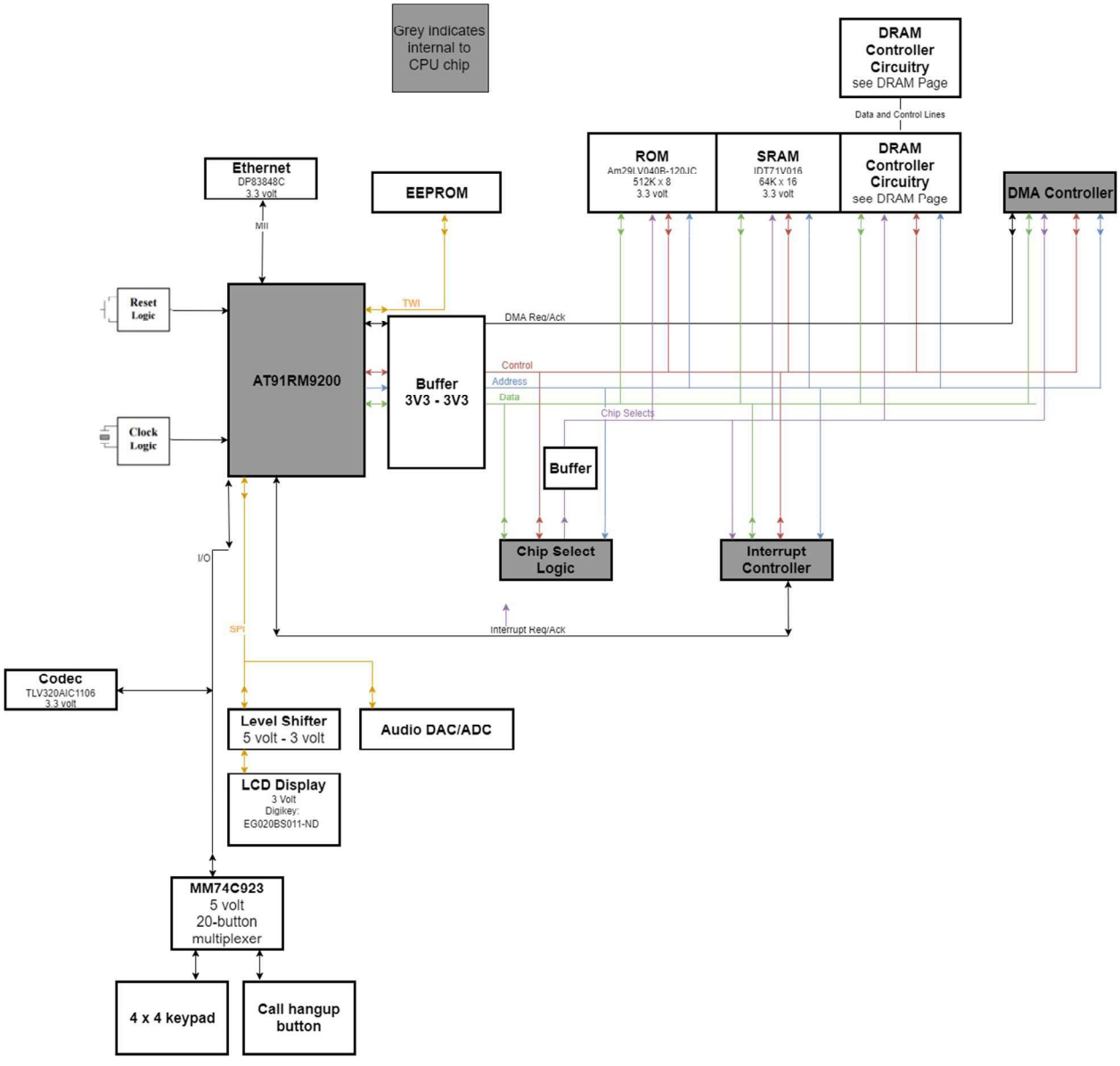
ROM_READ


0ps
50ns
100ns
150ns
CLK
t_SMC3
A
[15..0]
tBuffer
tBuffer
AB
[15..0]
t_SMC4
t_SMC4
NCS
tBuffer
tBuffer
NCSB
t_SMC37
t_SMC30
t_SMC32
NRD
tBuffer
tBuffer
NRDB
tOE
tCE
tACC
tOHZ
DATA
[15..0]
tBuffer
t_SMC40
t_SMC42
t_SMC41

Overview

Processor	Atmel AT91RM9200
Input power	8V-12V DC 5.5mm barrel jack
Display	128x32
Headphone Jack	3.5mm mono audio jack
Microphone Jack	3.5mm
Keypad	4x4 button keypad

Block Diagram



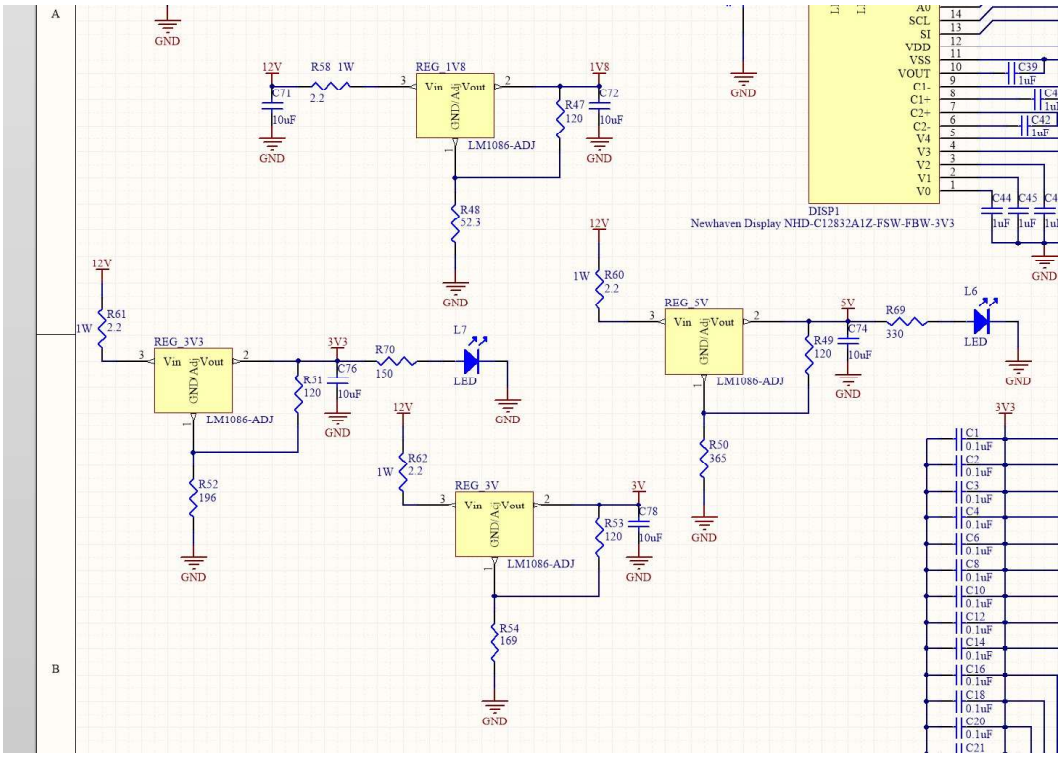
Power Circuitry

The system takes ~8V-12V DC input. The upper bound is due to thermal dissipation constraints of the linear regulator heatsinks. There are four power rails on this board.

Rail	Uses
5V	Reset circuit, DRAM, CPLD
3V3	Audio, SRAM, ROM, CPU
3V	Display
1V8	CPU Core logic

Linear Regulators

The system uses LM1086 linear regulators with heatsinks attached.

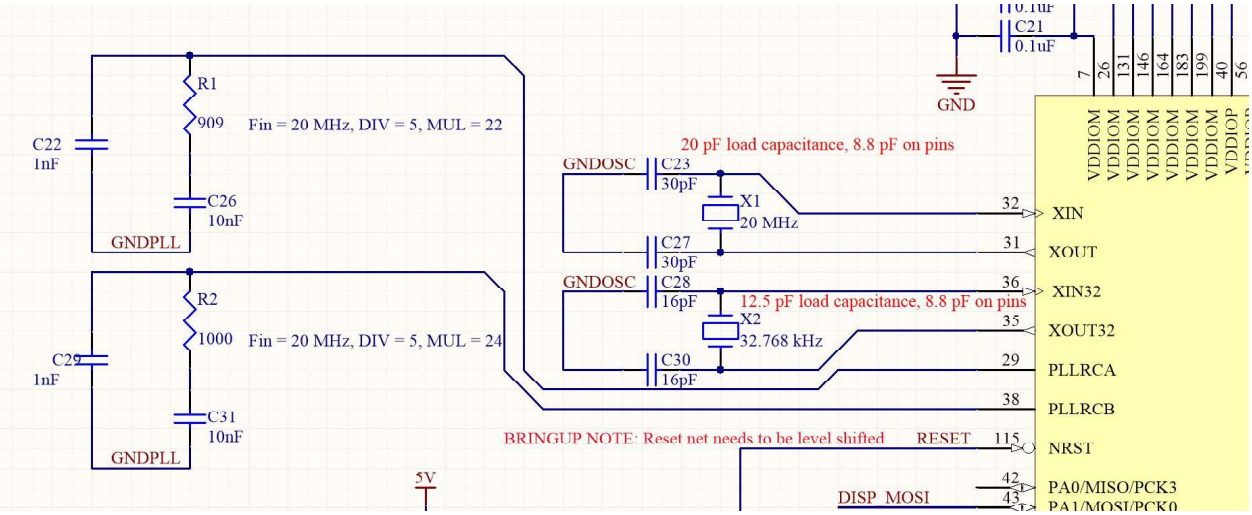


Linear regulators

CPU Support Circuitry

Clocks

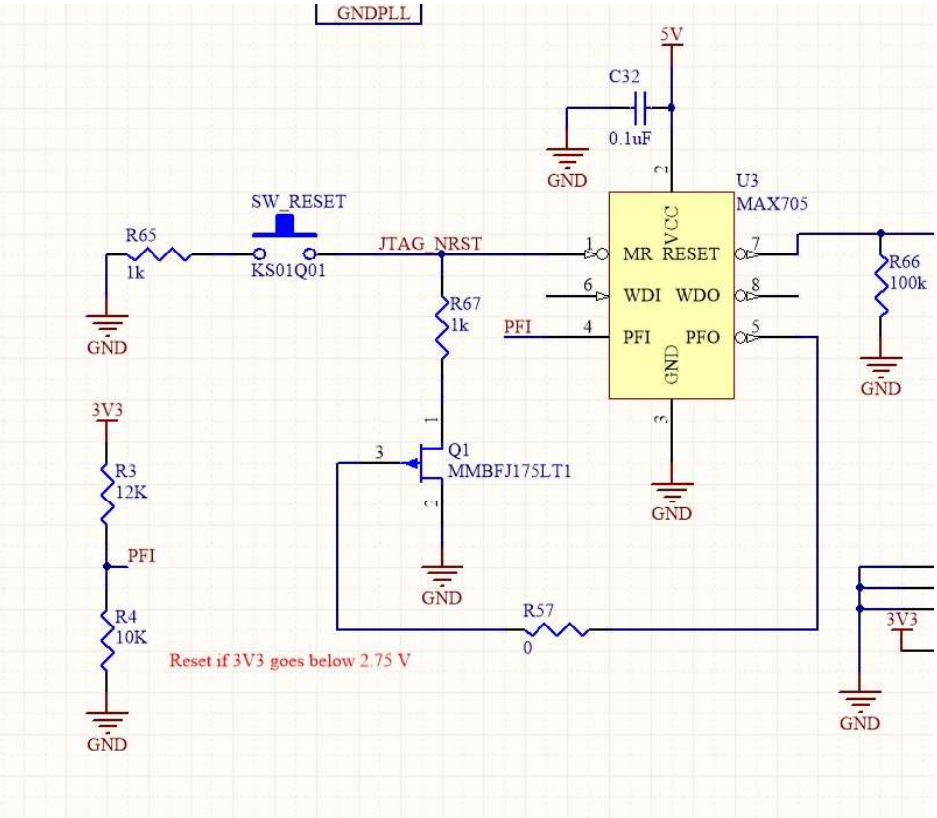
The CPU uses a 32.768 kHz slow clock and a 20 MHz main clock crystal. The main clock is scaled using a PLL to 44 MHz.



Clocks and PLL circuits

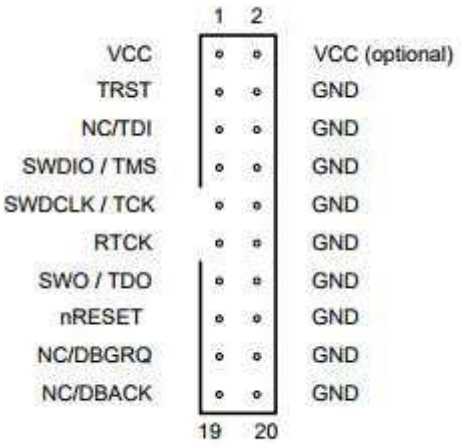
Reset

The reset circuitry uses a MAX705 chip in the standard configuration. The 5-volt reset line output is scaled using a resistor divider for devices that need 3.3V input.



JTAG

This board has a JTAG interface routed and populated for the purposes of debugging. It is a 20-pin legacy ARM IDC 0.1” pitch JTAG connector with the following pinout (see http://infocenter.arm.com/help/topic/com.arm.doc.faqs/attached/13634/cortex_debug_connectors.pdf):

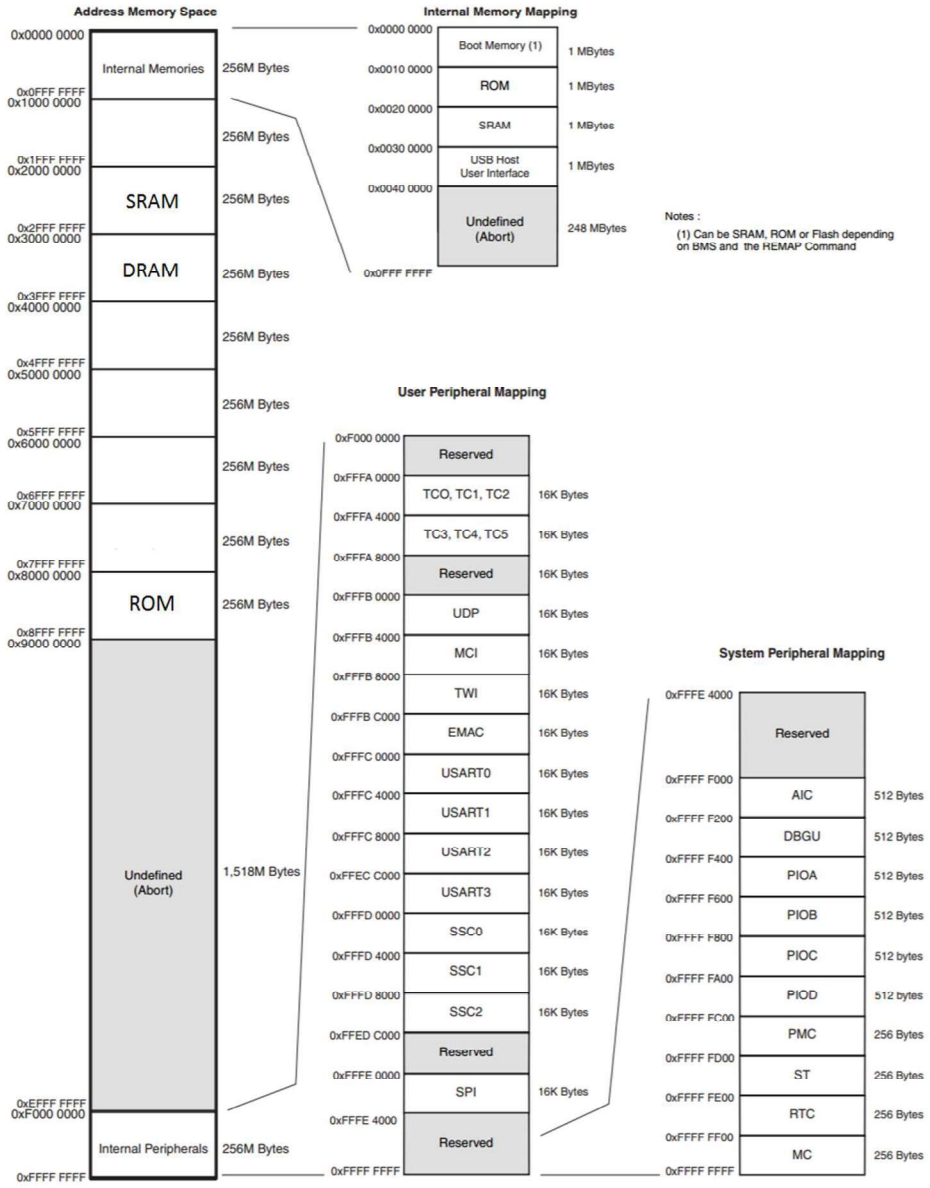


Memory

All memory access is done through 74LVT16245 buffers.

8. Memories

Figure 8-1. AT91RM9200 Memory Mapping



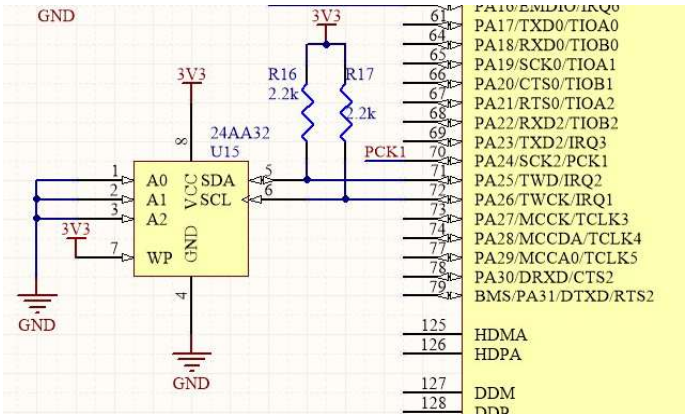
Memory map

Memory Testing

When the system boots, all of the memory in the SRAM and DRAM is checked for errors before the code is copied from ROM to SRAM for program operation. This is done in the memtest.s file. The test works by writing a few sequences of numbers that do not repeat with any 2^n periodicity, and reading back the data and comparing it against the expected sequence. This test seems to do a good job at detecting rare memory integrity issues, but it does not test for DRAM refresh being fast enough to prevent DRAM bit rot.

EEPROM Bootloader

The system uses a 24AA32 over TWI for the bootloader. This bootloader initializes the ARM CPU modes, checks for SRAM and DRAM integrity, and then copies ROM to SRAM. Future possible improvements could be to check a hash of the ROM values against a hash stored in ROM to verify ROM integrity.



ROM

This board uses a 29LV040 chip for ROM. This chip has an address bus width of 19 bits, and a data bus width of 8 bits. It is on chip select 7. See schematic in Appendix.

SRAM

This board uses a IDT71V016 chip for SRAM. This chip has an address bus width of 16 bits, and a data bus width of 16 bits. It is on chip select 1. See schematic in Appendix.

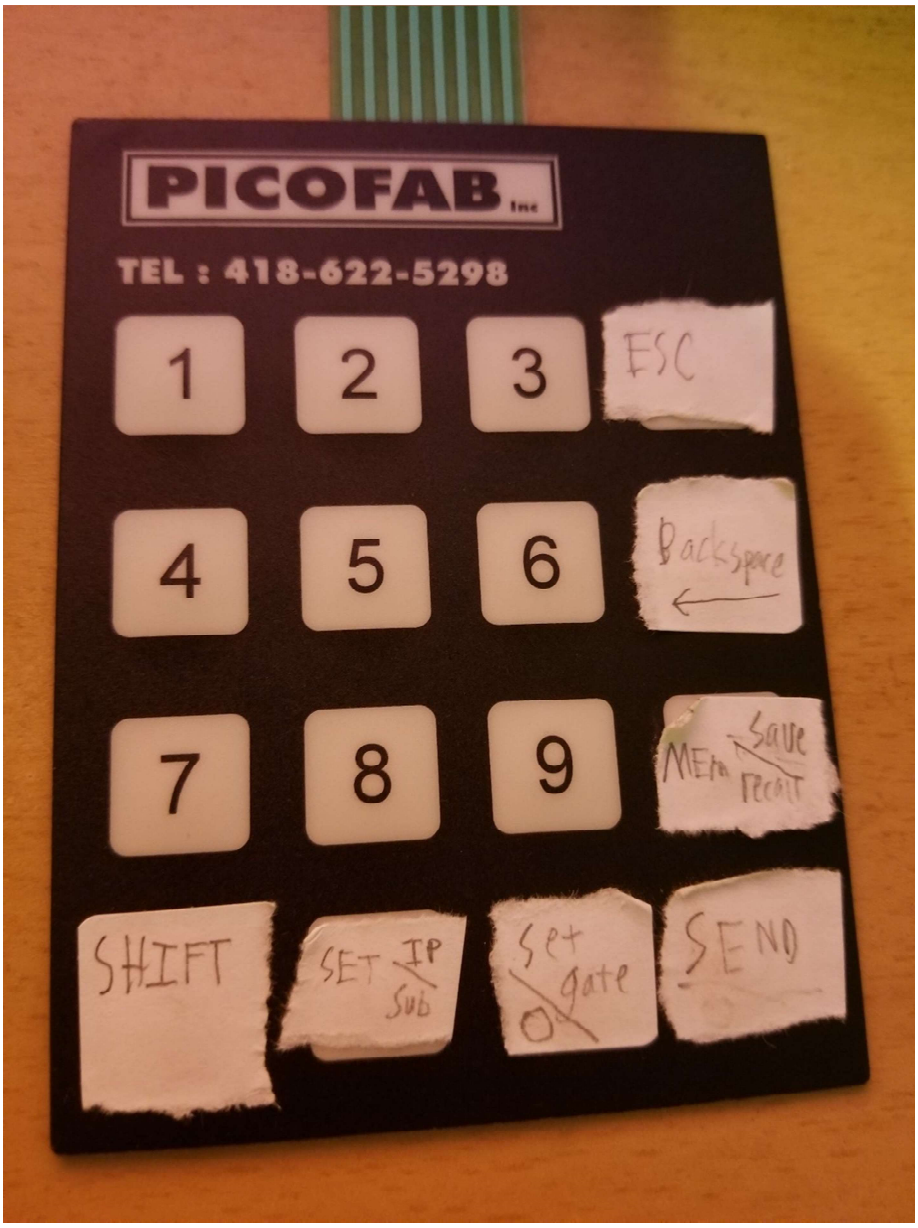
DRAM

This board uses a socket mount DRAM module. The DRAM controller is implemented in a CPLD. The address lines are multiplexed to the DRAM via U19 and U21, which are two buffers that can be toggle high-Z and enabled by the CPLD DRAM controller. The data lines are connected to the CPU data bus via a bidirectional buffer that is enabled when the DRAM chip select goes low. The directionality of the buffer is set by the NRD signal.

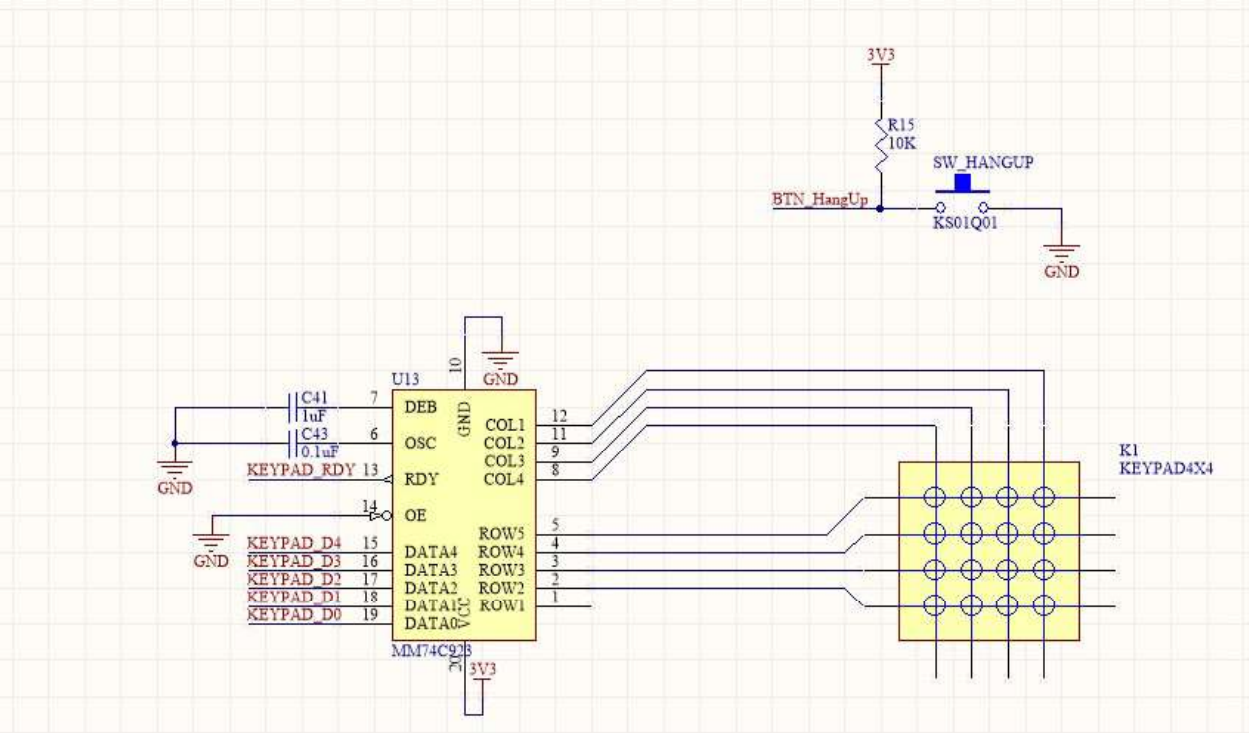
The DRAM has an effective address bus width of 20 bits, and a data bus width of 8 bits. The DRAM is on chip select 2. See Schematic in Appendix.

User Interface Components

Keypad

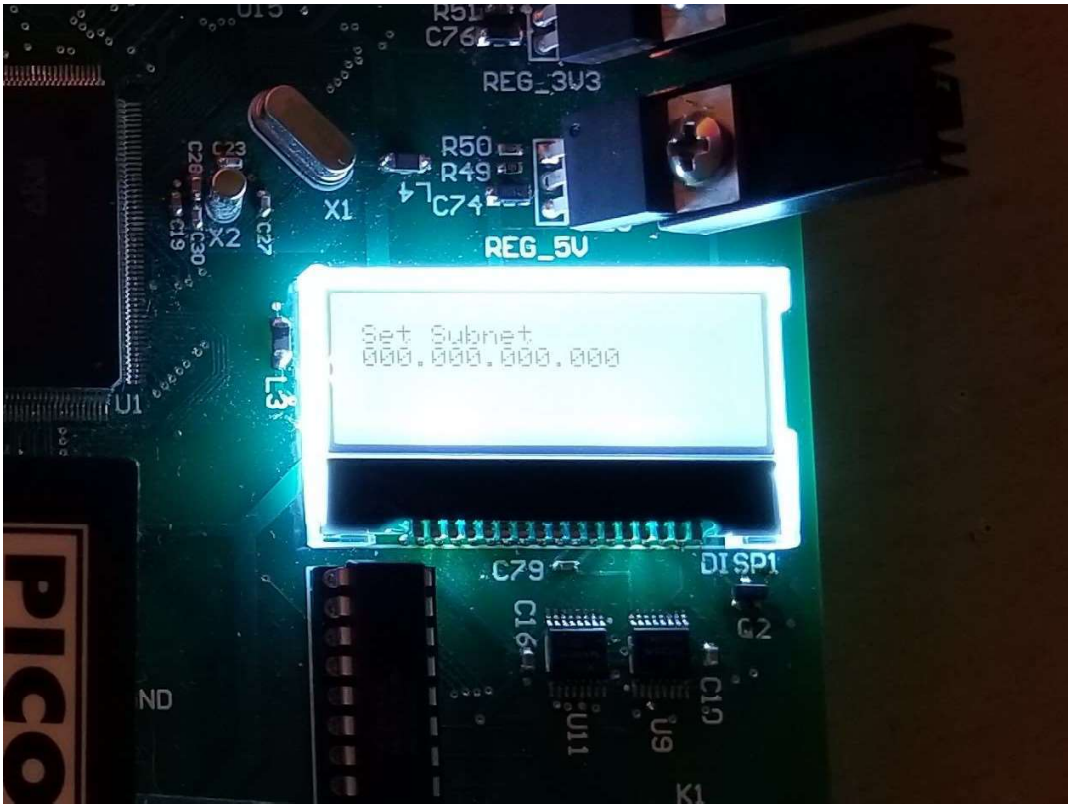


The keypad is used for user input into the device. It is a 4x4 keypad, and since more inputs are needed than are available, a shift key is used to toggle the function of a few keys. The keypad uses a MM74C923 chip for debouncing key presses. Also, on the test board, the hangup button was connected into the ROW1 channel of the chip, to simplify code. The design implemented in schematic works as well, but if the hangup button remains a button that is pressed when it is needed, rather than being a button that is always pressed when the phone is on the hook, then the hangup button can be connected to the debouncer chip. The debouncer chip can only debounce one key at a time, so if the hangup button is pressed while the user wants to press other buttons, the system would not respond to the new button presses, hence why the hangup button was kept separate in the schematic right now.



The above schematic shows the design as routed on the PCB. This design is fully functional, but on the test board, the hangup button was wired into row 1 of debouncer chip for code simplicity.

Display



The display is a NHDC12832 128x32 display. In software, the display is divided into 4 rows. Only the top two rows are used.

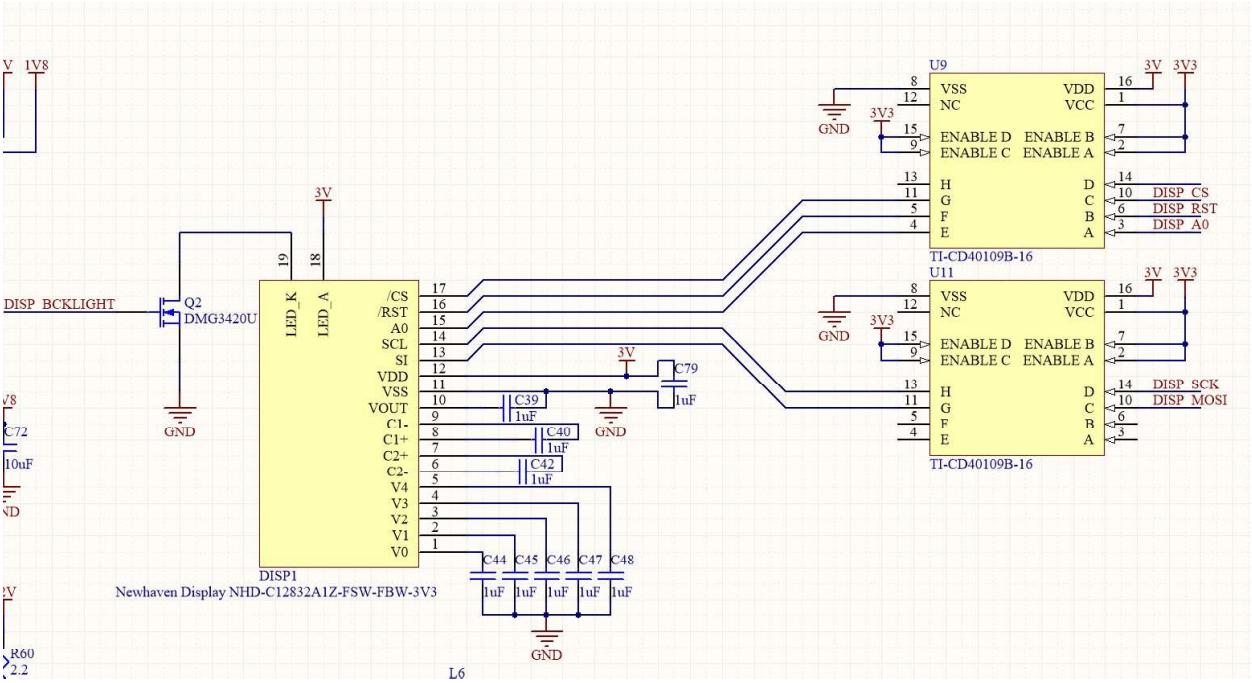
The top row displays the current menu the system is in, or the status of the system if a call is in progress.

The 2nd row displays auxiliary information for a menu, such as the IP address being set, the subnet mask, or the memory location for recall.

Data is streamed to the display using SPI, with the data transfer to the SPI peripheral done using DMA.

The display uses 3 volt power, while the CPU uses 3.3 volt, so level shifters are used.

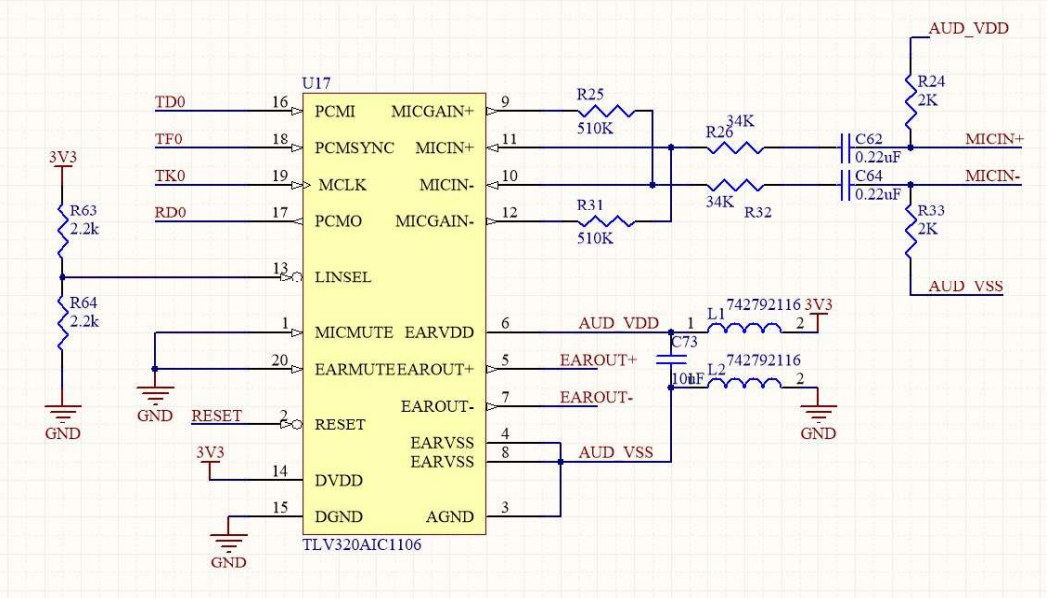
A backlight is available on the display, and is toggled via pin PA5 on the CPU.



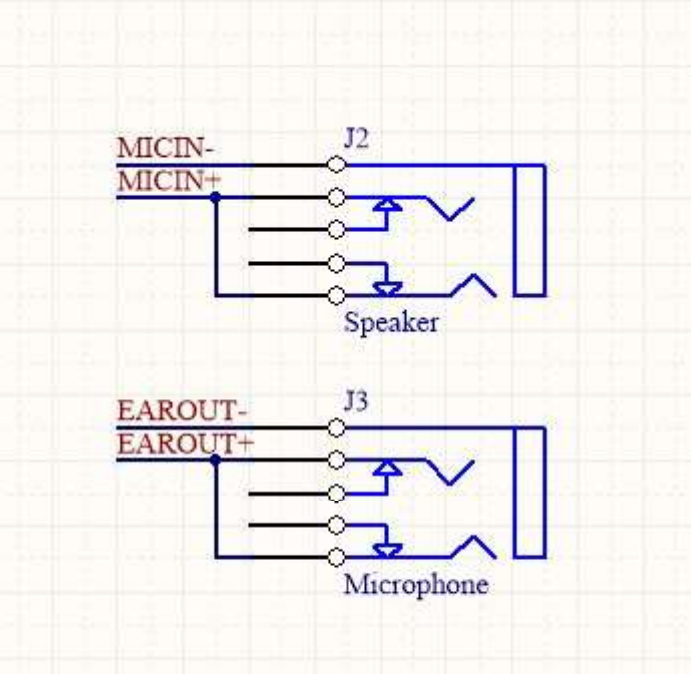
Display and level shifters for display

Audio

The audio circuitry uses the TLV320AIC1106 chip. Depending on whether companding or linear serial stream is desired, R63 or R64 can be stuffed. Currently, the code base only supports linear mode, which corresponds to R64 being stuffed. Data is sent and received from the audio chip via a synchronous serial controller on the ARM chip and a DMA engine on the ARM chip.



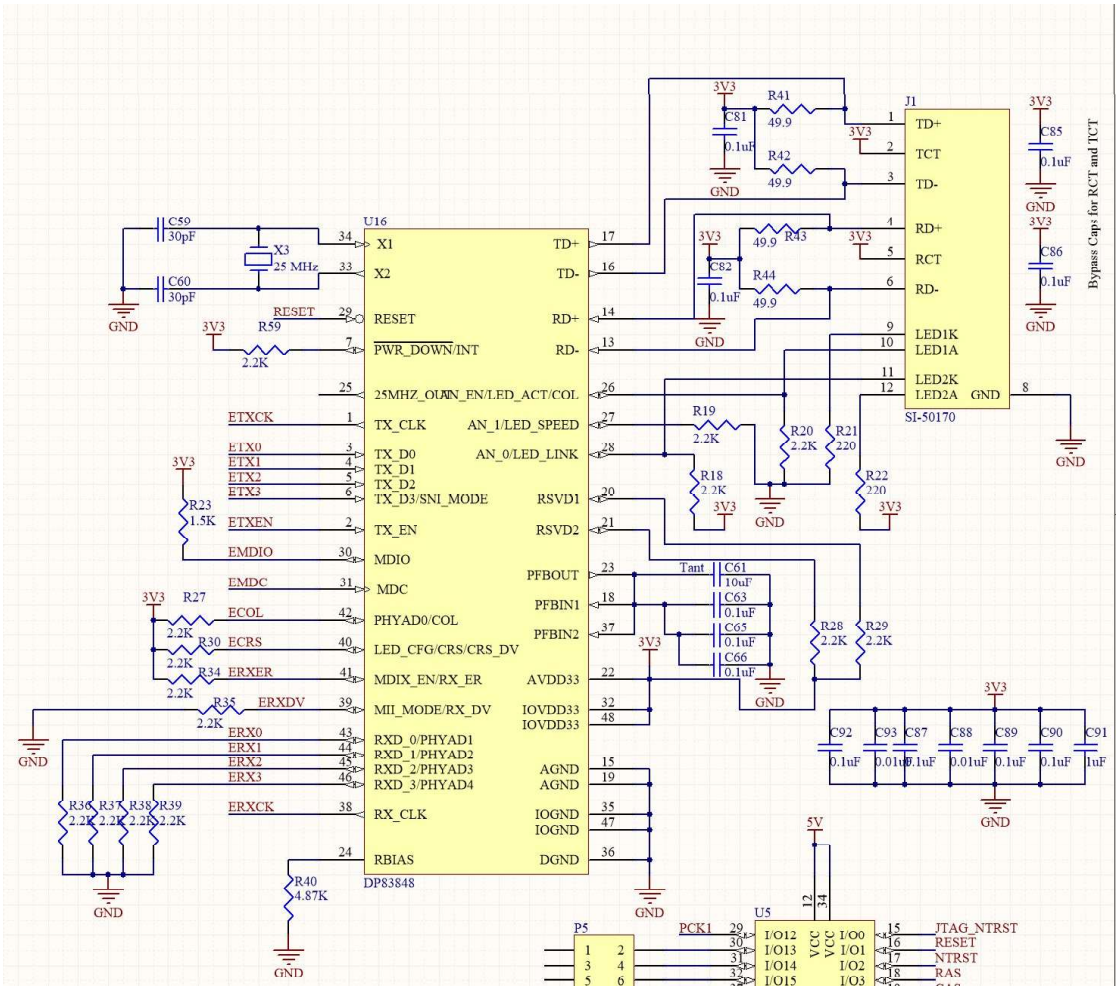
Audio Circuitry



Audio output

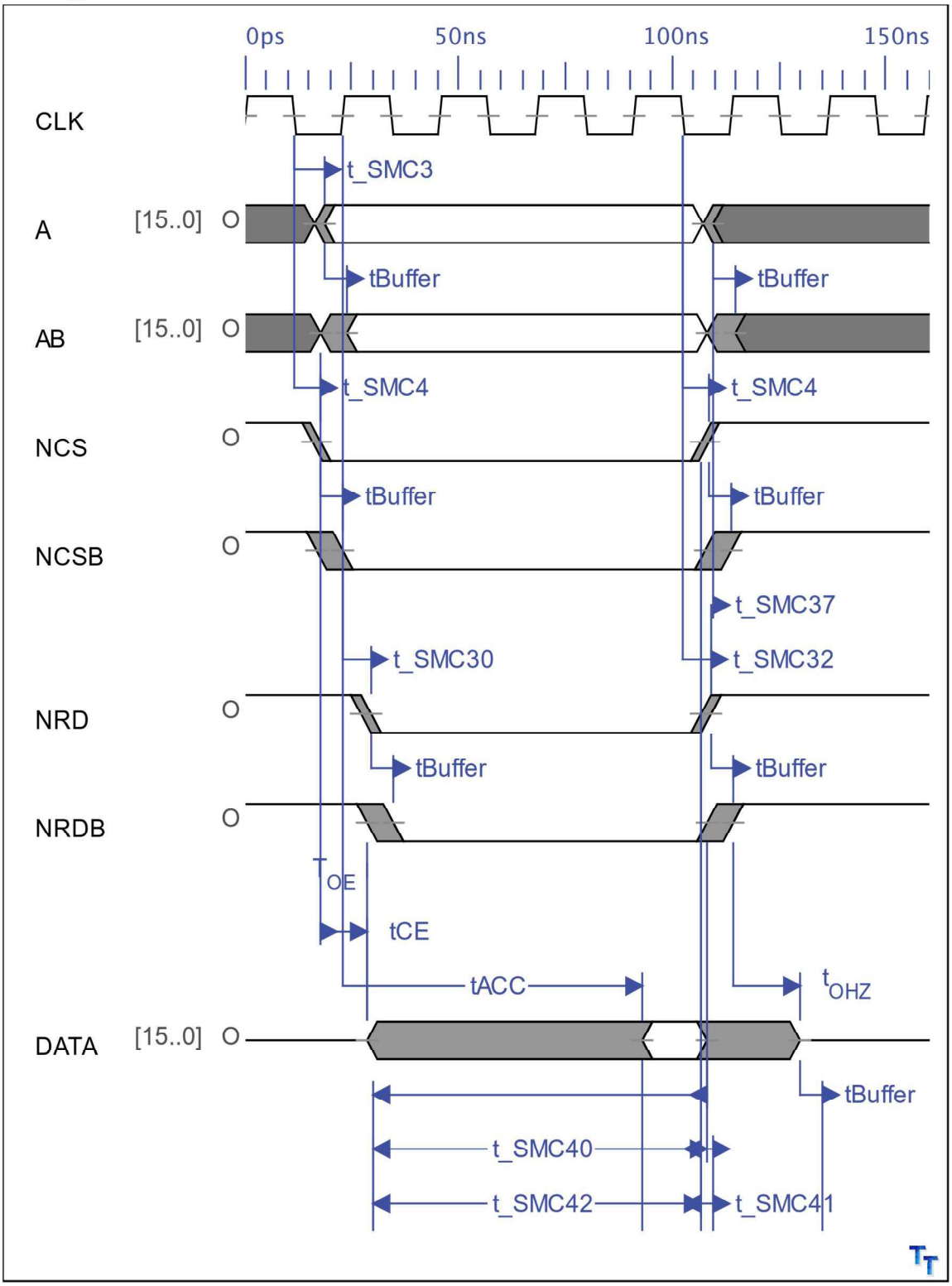
Ethernet

The ethernet circuitry in the schematic is untested, and may not work. It is designed to operate in either 10Base-T or 100Base-T, half or full duplex.

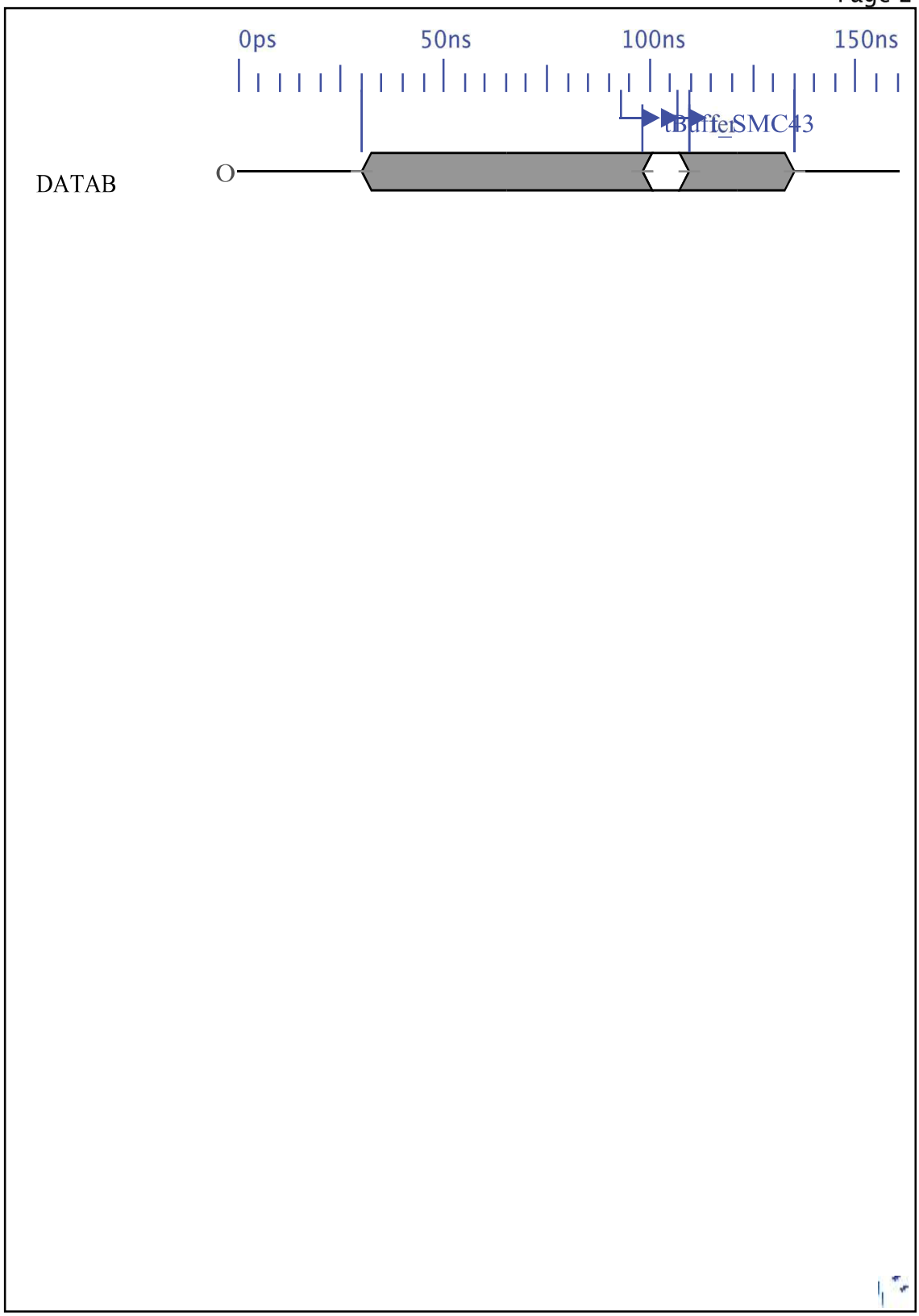

U16
DP83848
J1
SI-50170
U5
P5
X3
25 MHz
C59
30pF
C60
30pF
RESET
R59
2.2K
PWR_DOWN/INT
X1
X2
TD+
TD-
RD+
RD-
25MHZ_OUT
AN_EN/LED_ACT/COL
AN_1/LED_SPEED
AN_0/LED_LINK
RSVD1
RSVD2
PFBOUT
PFBIN1
PFBIN2
AVDD33
IOVDD33
IOVDD33
AGND
AGND
IOGND
IOGND
DGND
TX_CLK
TX_D0
TX_D1
TX_D2
TX_D3/SNI_MODE
TX_EN
MDIO
MDC
PHYAD0/COL
LED_CFG/CRS/CRS_DV
MDIX_EN/RX_ER
MII_MODE/RX_DV
RXD_0/PHYAD1
RXD_1/PHYAD2
RXD_2/PHYAD3
RXD_3/PHYAD4
RX_CLK
RBIAS
ETXCK
ETX0
ETX1
ETX2
ETX3
ETXEN
EMDIO
EMDC
ECOL
ECRS
ERXER
ERXDV
ERX0
ERX1
ERX2
ERX3
ERXCK
R23
1.5K
R27
2.2K
R30
2.2K
R34
2.2K
R35
2.2K
R36
2.2K
R37
2.2K
R38
2.2K
R39
2.2K
R40
4.87K
R41
49.9
R42
49.9
R43
49.9
R44
49.9
R19
2.2K
R18
2.2K
R20
2.2K
R21
220
R22
220
R28
2.2K
R29
2.2K
C81
0.1uF
C82
0.1uF
C85
0.1uF
C86
0.1uF
Tant
C61
10uF
C63
0.1uF
C65
0.1uF
C66
0.1uF
C92
0.1uF
C93
0.01uF
C87
0.1uF
C88
0.01uF
C89
0.1uF
C90
0.1uF
C91
1uF
TD+
TCT
TD-
RD+
RCT
RD-
LED1K
LED1A
LED2K
LED2A
GND
Bypass Caps for RCT and TCT
PCK1
I/O12
I/O13
I/O14
I/O15
VCC
VCC
I/O0
I/O1
I/O2
I/O3
JTAG_NTRST
RESET
NTRST
RAS
3V3
5V
GND

Appendix A (ROM Timing)

ROM_READ Page 1



C:\Users\will\GitReps\EE52\ROM_READ.tdml


0ps
50ns
100ns
150ns
tBuffer_SMC43
DATAB
O

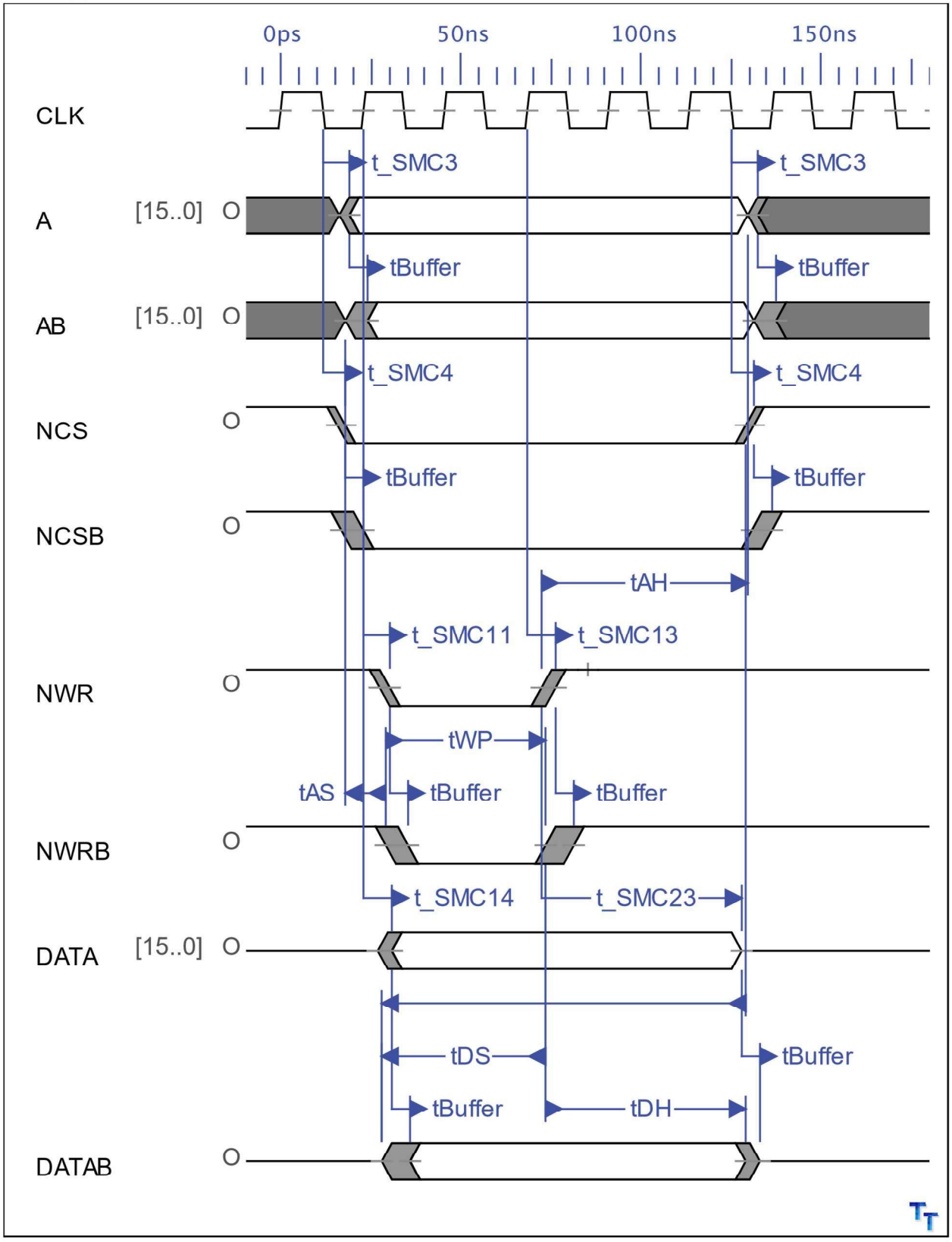
General Data

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
t_SMC3	MCK Falling to A1-A25 valid	to be defined	4.9ns	7.5ns	
t_SMC4	MCK Falling to Chip Select	to be defined	4.3ns	6.5ns	
tBuffer		to be defined	1.3ns	5.1ns	
t_SMC1		to be defined			
t_SMC30	MCK Rising to NRD Active	to be defined	4.7ns	7.0ns	
tACC	Address Access time	defined by memory	0ps	70ns	
tCE	Chip Select Access time	defined by memory	0ns	70ns	
T_{OE}	Output enable access time	defined by memory	0ns	30ns	
t_SMC32		to be defined	4.5ns	6.8ns	
t_SMC40	Data Setup time before NRD	to be defined	7.5ns		
t_SMC4	MCK falling to CHip Select Change	to be defined	4.3ns	6.5ns	
t_SMC41	Data Hold time after NRD rising	to be defined	-3.4ns		
t_SMC37	NRD High to A1-A25 change	to be defined	0.3ns	0.6ns	
t_{OHZ}	Output Enable High to Output in High-Z	defined by memory	0ns	16ns	
t_SMC42	Data Setup before NCS High	to be defined	7.3ns		
t_SMC43	Data Hold after NCS High	to be defined	-3.2ns		

Notes:

Device: AM29LV040

Results:
1 wait states



C:\Users\will\GitReps\EE52\ROM_WRITE.tdml

General Data

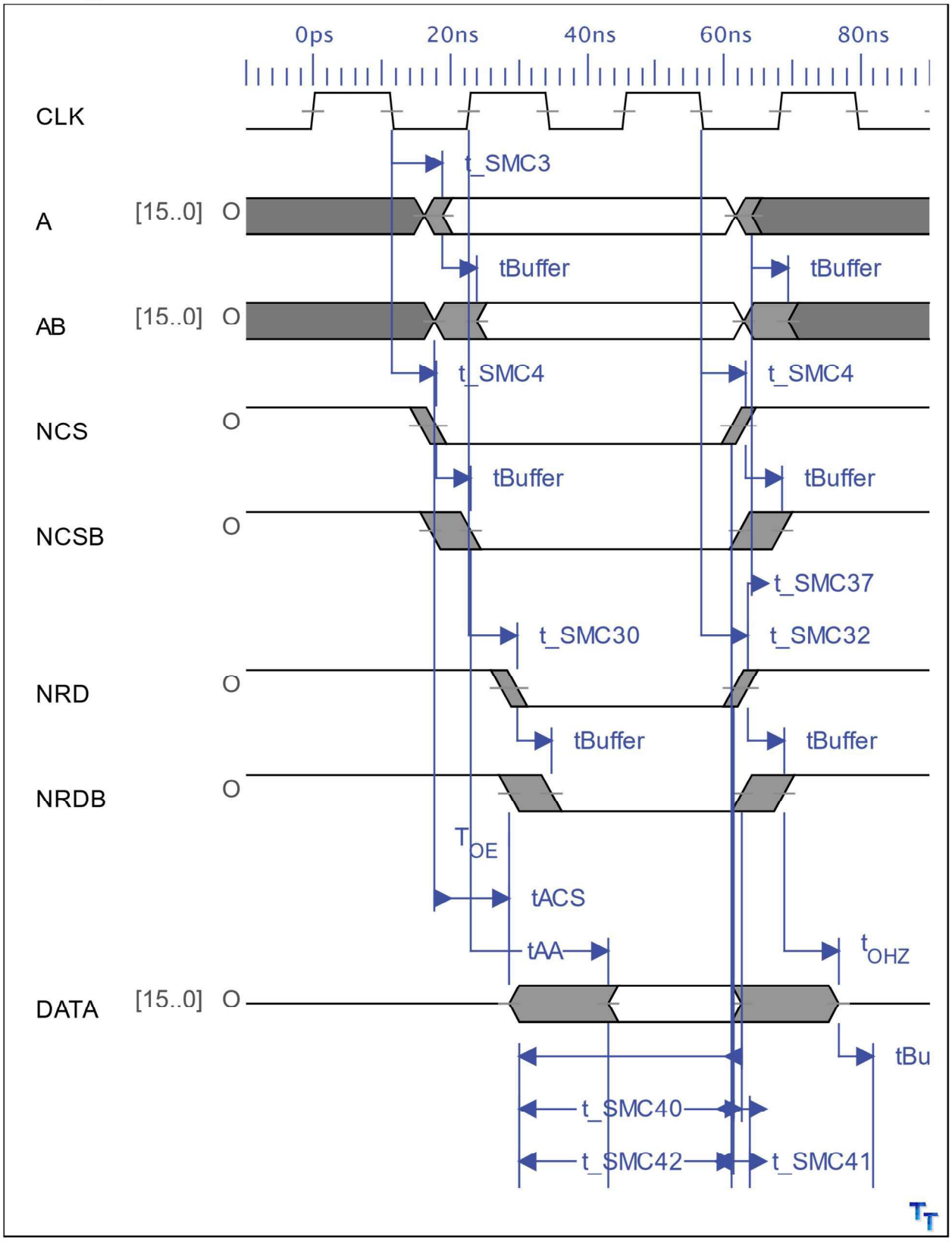
SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
t_SMC3	MCK Falling to A1-A25 valid	to be defined	4.9ns	7.5ns	
tBuffer		to be defined	1.3ns	5.1ns	
t_SMC14	MCK Rising to D0-D15 Valid	to be defined	4.1ns	7.9ns	
t_SMC4	MCK falling to CHip Select Change	to be defined	4.3ns	6.5ns	
tAS	Address Setup to End of Write	to be defined	0ns		
tDS	Data Setup Time	to be defined	35ns		
tDH	Data Hold Time	to be defined	0ns		
tAH	Address Hold from End of Write	to be defined	45ns		
tWP	Write Pulse	to be defined	35ns		
t_SMC11	MCK Rising to NWR Active (with wait states)	to be defined	4.8ns	7.2ns	
t_SMC13	MCK Rising to NWR Inactive	to be defined	4.1ns	7.9ns	
t_SMC23	MCK Rising to Data Out Invalid (1 hold state)	to be defined	55.7ns		
t_SMC17	NWR High to A1-A25 Change	to be defined	3.3ns		

Notes:

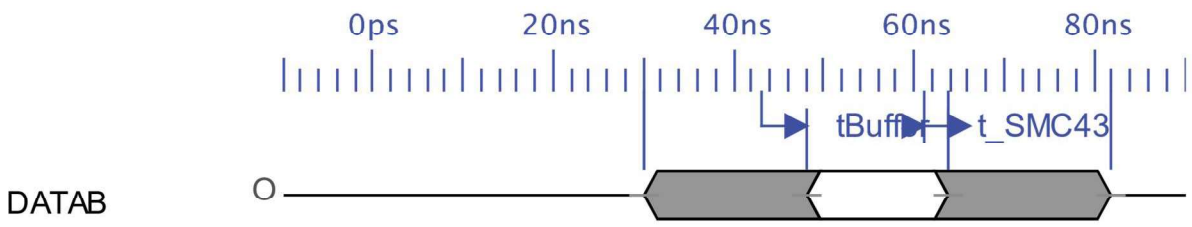
Results:
0 setup
2 wait state (pulse length)
2.5 hold parameter

Appendix B (SRAM Timing)

SRAM_READ



C:\Users\will\GitReps\EE52\SRAM_READ.tdml


0ps
20ns
40ns
60ns
80ns
tBuffer
t_SMC43
DATAB
O

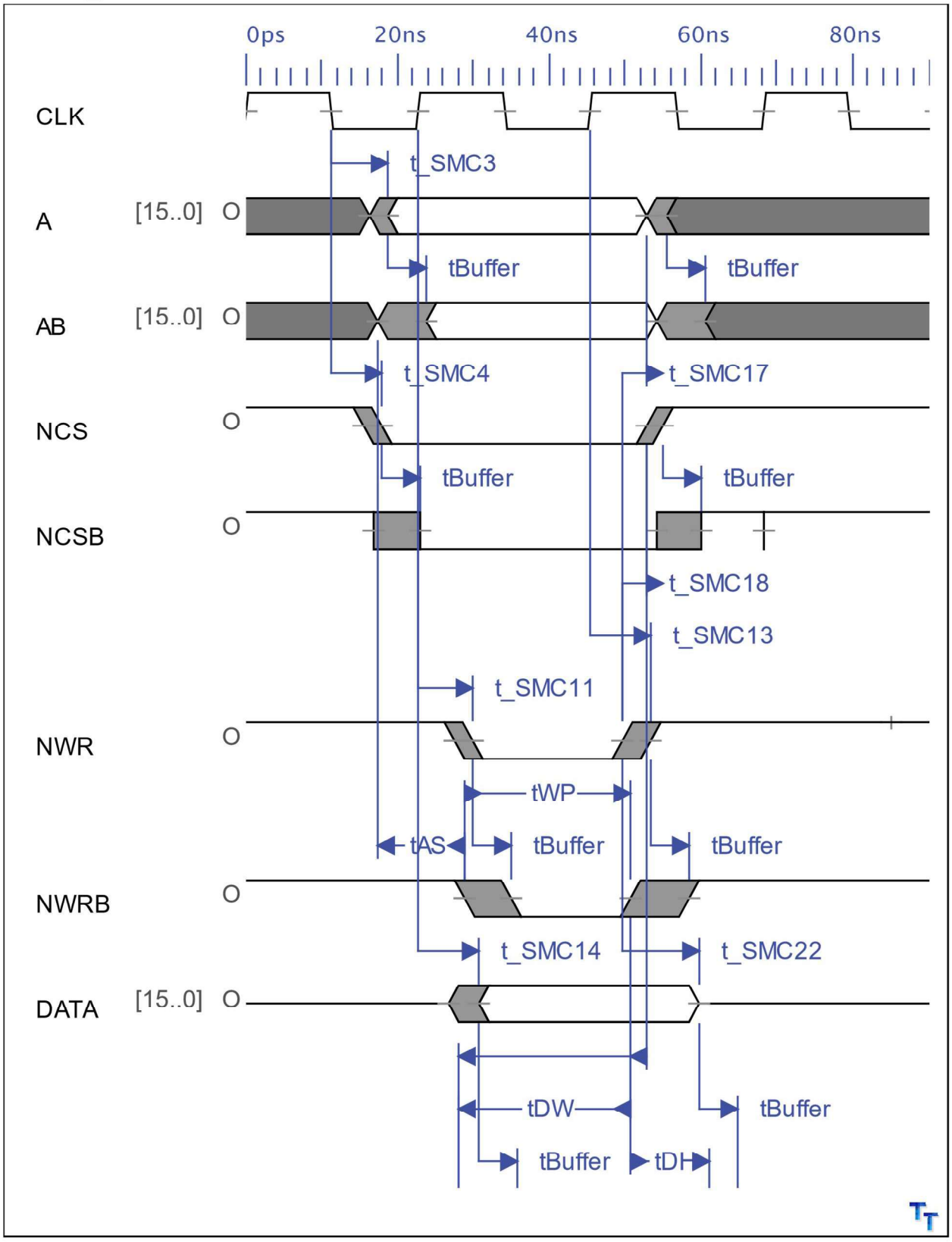
General Data

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
t_SMC3	MCK Falling to A1-A25 valid	to be defined	4.9ns	7.5ns	
t_SMC4	MCK Falling to Chip Select	to be defined	4.3ns	6.5ns	
tBuffer		to be defined	1.3ns	5.1ns	
t_SMC1		to be defined			
t_SMC30	MCK Rising to NRD Active	to be defined	4.7ns	7.0ns	
tAA	Address Access time	defined by memory	0ps	20ns	
tACS	Chip Select Access time	defined by memory	0ns	20ns	
T_{OE}	Output enable access time	defined by memory	0ns	8ns	
t_SMC32	MCK falling to NRD inactive	to be defined	4.5ns	6.8ns	
t_SMC40	Data Setup time before NRD	to be defined	7.5ns		
t_SMC4	MCK falling to CHip Select Change	to be defined	4.3ns	6.5ns	
t_SMC41	Data Hold time after NRD rising	to be defined	-3.4ns		
t_SMC37	NRD High to A1-A25 change	to be defined	0.3ns	0.6ns	
t_{OHZ}	Output Enable High to Output in High-Z	to be defined	0ns	8ns	
t_SMC42	Data Setup before NCS High	to be defined	7.3ns		
t_SMC43	Data Hold after NCS High	to be defined	-3.2ns		

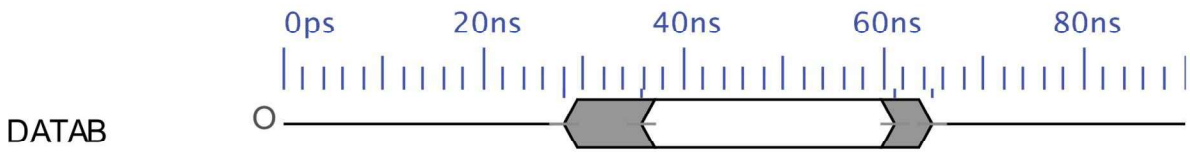
Notes:

Results:
1 wait states

SRAM_WRITE



C:\Users\will\GitReps\EE52\SRAM_WRITE.tdml


0ps
20ns
40ns
60ns
80ns
DATAB

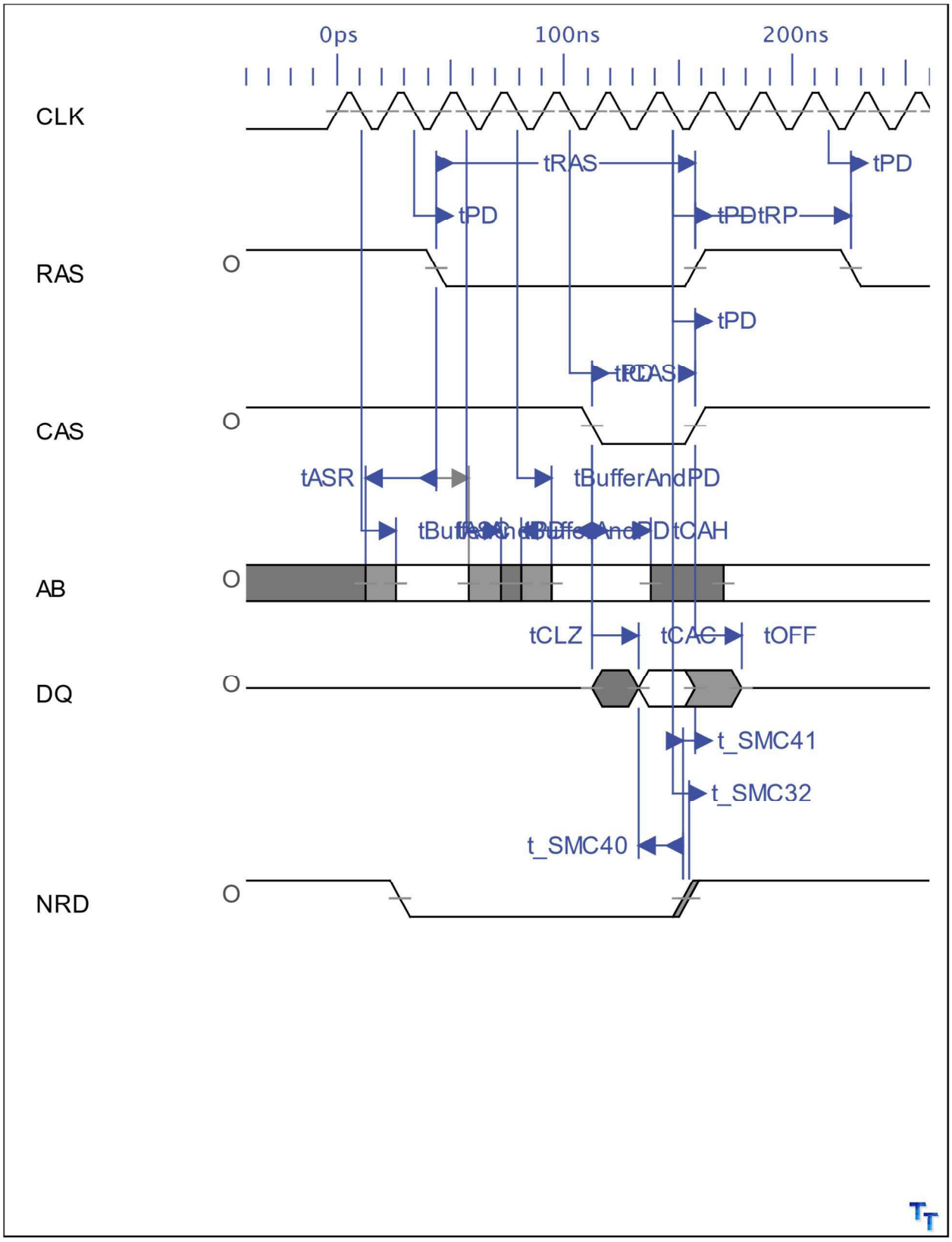
General Data

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
t_SMC3	MCK Falling to A1-A25 valid	to be defined	4.9ns	7.5ns	
tBuffer		to be defined	1.3ns	5.1ns	
t_SMC14	MCK Rising to D0-D15 Valid	to be defined	4.1ns	7.9ns	
t_SMC4	MCK falling to CHip Select Change	to be defined	4.3ns	6.5ns	
tAS	Address Setup to End of Write	to be defined	0ns		
tDW	Data Setup Time	to be defined	9ns		
tDH	Data Hold Time	to be defined	0ns		
tWR	Address Hold from End of Write	to be defined	0ns		
tWP	Write Pulse	to be defined	12ns		
t_SMC11	MCK Rising to NWR Active (with wait states)	to be defined	4.8ns	7.2ns	
t_SMC13	MCK Rising to NWR Inactive	to be defined	4.1ns	7.9ns	
t_SMC22	MCK Rising to Data Out Invalid	to be defined	10.2ns		
t_SMC17	NWR High to A1-A25 Change	to be defined	3.3ns		
t_SMC18	NWR High to Chip Select Inactive	to be defined	3.3ns		

Notes:

Results:
0 setup
1 wait state (pulse length)
0 hold parameter

Appendix C (DRAM Timing)



C:\Users\will\GitReps\EE52\DRAMREAD.tdml

General Data

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tASR	Row-address setup time before RAS low	to be defined	0ns		
tASC	Column-address setup time before CAS low	to be defined	0ns		
tDS	Data setup time	to be defined	0ns		
tRCS	Read setup time before CAS low	to be defined			
tCWL	W low setup time before CAS	to be defined	20ns		
tRWL	W low setup time before RAS	to be defined			
tWCS	W low setup time before CAS	to be defined	0ns		
tWSR	W high setup time (CAS-before-RAS refresh only)	to be defined	10ns		
tCAH	Column-address hold time after CAS low	to be defined	15ns		
tDHR	Data hold time after RAS low	to be defined	60ns		
tDH	Data hold time	to be defined	15ns		
tAR	Column-address hold time after RAS low	to be defined	60ns		
tRAH	Row-address hold time after RAS low	to be defined	10ns		
tRCH	Read hold time after CAS high	to be defined	0ns		
tRRH	Read hold time after RAS high	to be defined	0ns		
tWCH	Write hold time after CAS low	to be defined	15ns		

tWCR	Write hold time after RAS low	to be defined	60ns		
tWHR	W high hold time (CAS-before-RAS refresh only)	to be defined	10ns		
tCHR	Delay time, RAS low to CAS high	to be defined	20ns		

General Data continued...

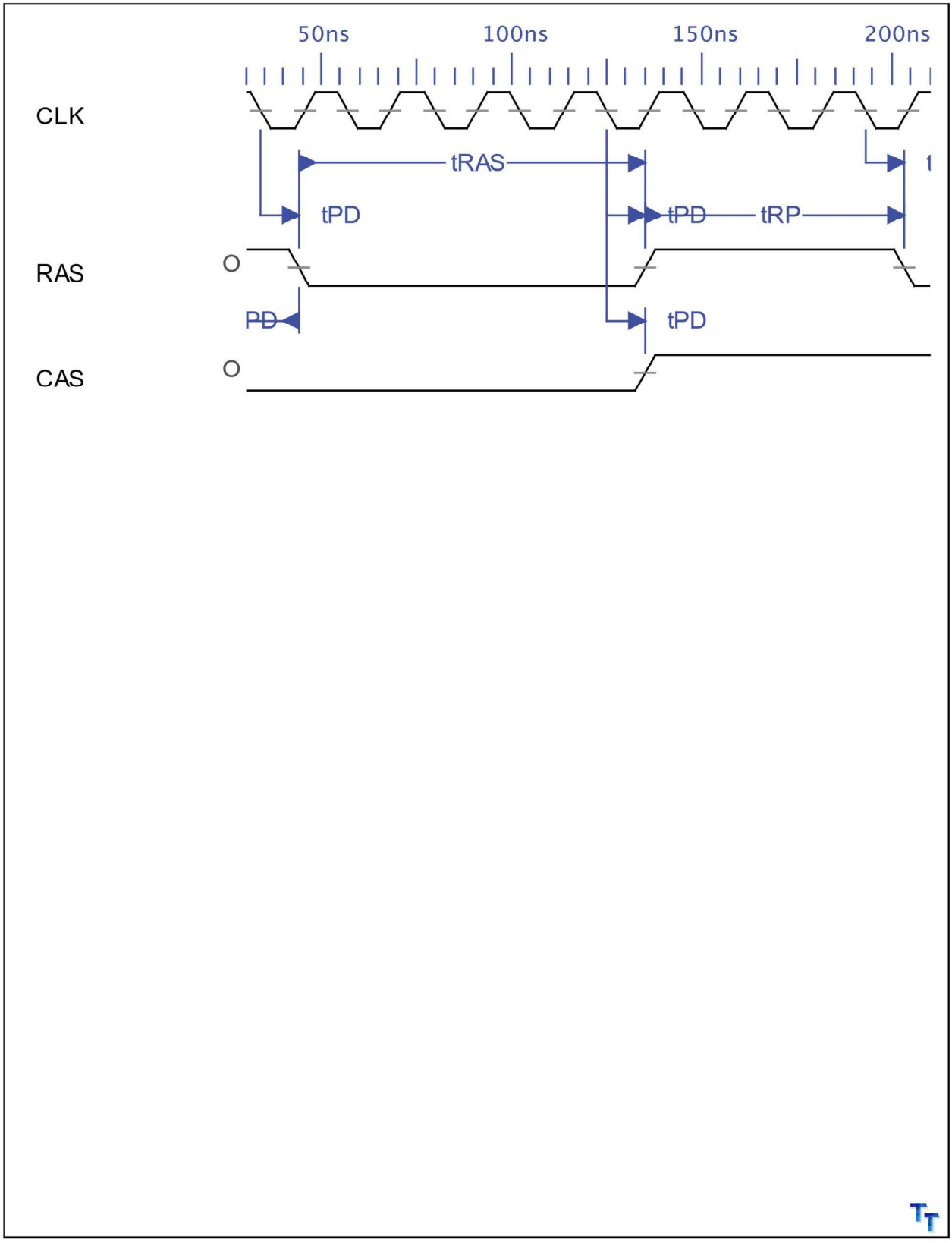
SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tCRP	Delay time, CAS high to RAS low	to be defined	0ns		
tCSH	Delay time, RAS low to CAS high	to be defined	80ns		
tCSR	Delay time, CAS low to RAS low	to be defined	10ns		
tRAD	Delay time, RAS low to column-address	to be defined	15ns	40ns	
tRAL	Delay time, column-address to RAS high	to be defined	40ns		
tCAL	Delay time, column-address to CAS high	to be defined	40ns		
tRCD	Delay time, RAS low to CAS low	to be defined	20ns	60ns	
tRPC	Delay time, RAS high to CAS low	to be defined	0ns		
tRSH	Delay time, CAS low to RAS high	to be defined	20ns		
tT	Transition time	to be defined	2ns	50ns	
tRAS	Non-page-mode pulse duration, RAS low	to be defined	80ns	10000ns	
tCAS	Pulse duration, CAS low	to be defined	20ns	10000ns	
tCLZ	CAS to output in low Z	to be defined	0ns		

tOFF	output disable time after CAS high	to be defined	0ns	20ns	
tBufferAndPD	Buffer propagation delays	to be defined	1.3ns	15.1ns	
tPD	CPLD worst case path	to be defined		10ns	
tCAC	Access time from CAS low	to be defined		20ns	
tRAC	Access time from RAS low	to be defined		80ns	
tRP	Pulse duration, RAS high	to be defined	60ns		

General Data continued...

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
t_SMC40	Data Setup time before NRD	to be defined	7.5ns		
t_SMC41	Data Hold time after NRD rising	to be defined	-3.4ns		

t_SMC32	MCK falling to NRD inactive	to be defined	4.5ns	6.8ns



C:\Users\will\GitReps\EE52\DRAMREFRESH.tdml

General Data

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tASR	Row-address setup time before RAS low	to be defined	0ns		
tASC	Column-address setup time before CAS low	to be defined	0ns		
tDS	Data setup time	to be defined	0ns		
tRCS	Read setup time before CAS low	to be defined			
tCWL	W low setup time before CAS	to be defined	20ns		
tRWL	W low setup time before RAS	to be defined			
tWCS	W low setup time before CAS	to be defined	0ns		
tWSR	W high setup time (CAS-before-RAS refresh only)	to be defined	10ns		
tCAH	Column-address hold time after CAS low	to be defined	15ns		
tDHR	Data hold time after RAS low	to be defined	60ns		
tDH	Data hold time	to be defined	15ns		
tAR	Column-address hold time after RAS low	to be defined	60ns		
tRAH	Row-address hold time after RAS low	to be defined	10ns		
tRCH	Read hold time after CAS high	to be defined	0ns		
tRRH	Read hold time after RAS high	to be defined	0ns		
tWCH	Write hold time after CAS low	to be defined	15ns		

tWCR	Write hold time after RAS low	to be defined	60ns		
tWHR	W high hold time (CAS-before-RAS refresh only)	to be defined	10ns		
tCHR	Delay time, RAS low to CAS high	to be defined	20ns		

General Data continued...

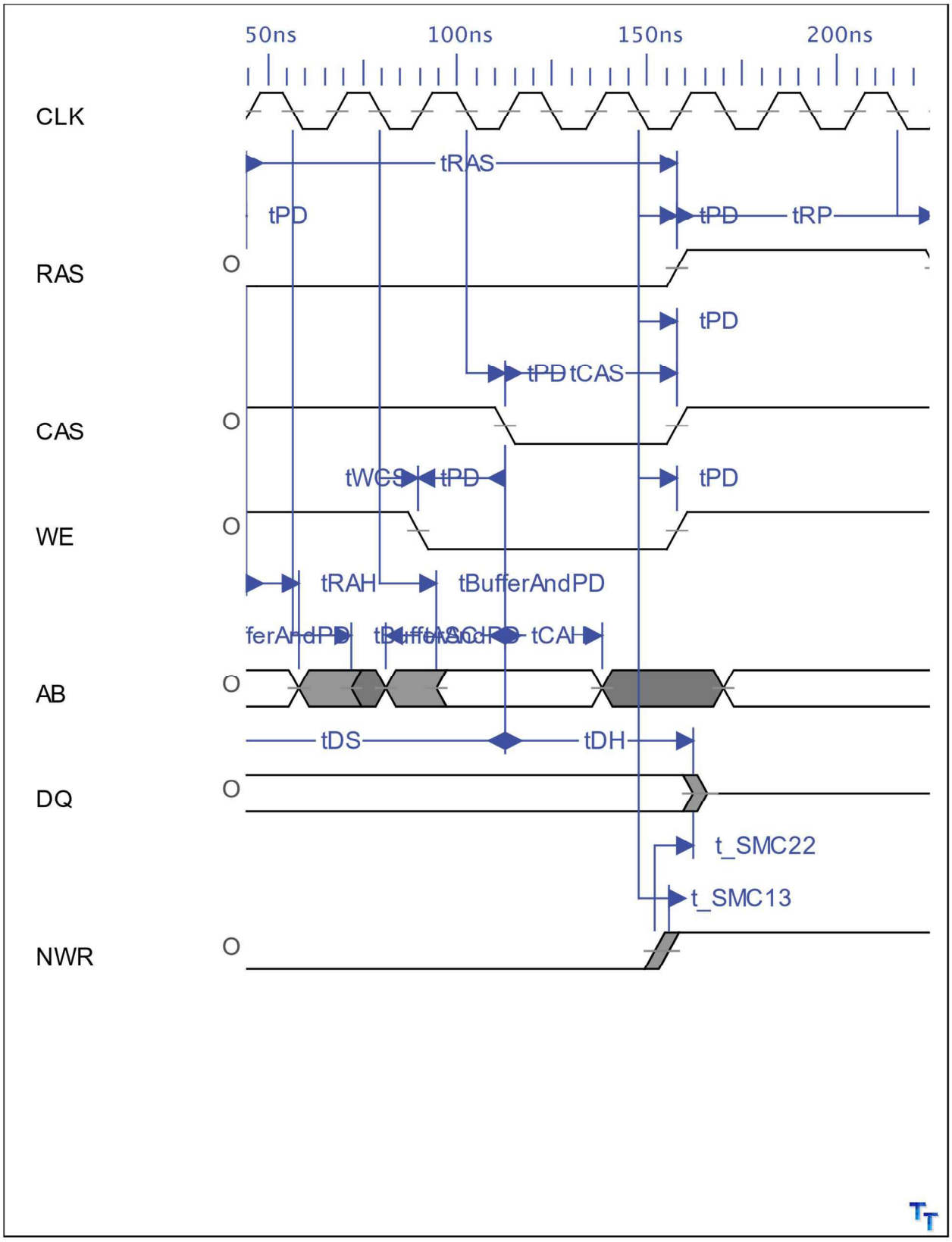
SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tCRP	Delay time, CAS high to RAS low	to be defined	0ns		
tCSH	Delay time, RAS low to CAS high	to be defined	80ns		
tCSR	Delay time, CAS low to RAS low	to be defined	10ns		
tRAD	Delay time, RAS low to column-address	to be defined	15ns	40ns	
tRAL	Delay time, column-address to RAS high	to be defined	40ns		
tCAL	Delay time, column-address to CAS high	to be defined	40ns		
tRCD	Delay time, RAS low to CAS low	to be defined	20ns	60ns	
tRPC	Delay time, RAS high to CAS low	to be defined	0ns		
tRSH	Delay time, CAS low to RAS high	to be defined	20ns		
tT	Transition time	to be defined	2ns	50ns	
tRAS	Non-page-mode pulse duration, RAS low	to be defined	80ns	10000ns	
tCAS	Pulse duration, CAS low	to be defined	20ns	10000ns	
tCLZ	CAS to output in low Z	to be defined	0ns		

tOFF	output disable time after CAS high	to be defined	0ns	20ns	
tBufferAndPD	Buffer propagation delays	to be defined	1.3ns	15.1ns	
tPD	CPLD worst case path	to be defined		10ns	
tCAC	Access time from CAS low	to be defined		20ns	
tRAC	Access time from RAS low	to be defined		80ns	
tRP	Pulse duration, RAS high	to be defined	60ns		

General Data continued...

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES

tCP	Pulse duration, CAS high	to be defined	10ns



C:\Users\will\GitReps\EE52\DRAMWRITE.tdml

General Data

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tASR	Row-address setup time before RAS low	to be defined	0ns		
tASC	Column-address setup time before CAS low	to be defined	0ns		
tDS	Data setup time	to be defined	0ns		
tRCS	Read setup time before CAS low	to be defined			
tCWL	W low setup time before CAS	to be defined	20ns		
tRWL	W low setup time before RAS	to be defined			
tWCS	W low setup time before CAS	to be defined	0ns		
tWSR	W high setup time (CAS-before-RAS refresh only)	to be defined	10ns		
tCAH	Column-address hold time after CAS low	to be defined	15ns		
tDHR	Data hold time after RAS low	to be defined	60ns		
tDH	Data hold time	to be defined	15ns		
tAR	Column-address hold time after RAS low	to be defined	60ns		
tRAH	Row-address hold time after RAS low	to be defined	10ns		
tRCH	Read hold time after CAS high	to be defined	0ns		
tRRH	Read hold time after RAS high	to be defined	0ns		
tWCH	Write hold time after CAS low	to be defined	15ns		

tWCR	Write hold time after RAS low	to be defined	60ns		
tWHR	W high hold time (CAS-before-RAS refresh only)	to be defined	10ns		
tCHR	Delay time, RAS low to CAS high	to be defined	20ns		

General Data continued...

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tCRP	Delay time, CAS high to RAS low	to be defined	0ns		
tCSH	Delay time, RAS low to CAS high	to be defined	80ns		
tCSR	Delay time, CAS low to RAS low	to be defined	10ns		
tRAD	Delay time, RAS low to column-address	to be defined	15ns	40ns	
tRAL	Delay time, column-address to RAS high	to be defined	40ns		
tCAL	Delay time, column-address to CAS high	to be defined	40ns		
tRCD	Delay time, RAS low to CAS low	to be defined	20ns	60ns	
tRPC	Delay time, RAS high to CAS low	to be defined	0ns		
tRSH	Delay time, CAS low to RAS high	to be defined	20ns		
tT	Transition time	to be defined	2ns	50ns	
tRAS	Non-page-mode pulse duration, RAS low	to be defined	80ns	10000ns	
tCAS	Pulse duration, CAS low	to be defined	20ns	10000ns	
tCLZ	CAS to output in low Z	to be defined	0ns		

tOFF	output disable time after CAS high	to be defined	0ns	20ns	
tBufferAndPD	Buffer propagation delays	to be defined	1.3ns	15.1ns	
tPD	CPLD worst case path	to be defined		10ns	
tCAC	Access time from CAS low	to be defined		20ns	
tRAC	Access time from RAS low	to be defined		80ns	
tRP	Pulse duration, RAS high	to be defined	60ns		

General Data continued...

SYMBOL	DEFINITION	DESCRIPTION	MIN	MAX	NOTES
tBuffer		to be defined	1.3ns	5.1ns	
t_SMC13	MCK Falling to NWR inactive	to be defined	4.1ns	7.9ns	

t_SMC22	MCK Rising to Data out invalid	to be defined	10.2ns

Appendix D (DRAM Abel Code)

```
MODULE DRAM

TITLE 'DRAM Interface - State Machine Version'

" DRAM Interface  EE52 Project


" Description:  Controls DRAM access


" Revision History:
" Will Werst   3/12/2017   Wrote code



" Pins


"      pin   1                  input   unused
"      pin   2                  input   unused
"      pin   3                  input   unused
"      pin   4                  input   unused
"      pin   5                  input   unused
"      pin   6                  input   unused
"      pin   7                  input   unused
"      pin   8                  input   unused
"      pin   9                  input   unused
"      pin   10                 input   unused
"      pin   11                 input   unused
"      pin   12                 input   unused
"      pin   13                 input   unused
"      pin   14                 input   unused
!JTAG_NTRST pin   15;           "JTAG NTRST input
!NReset pin  16;                "Reset input
!NTRST pin   17  ISTYPE 'com';  "TRST output
RAS    pin   18  ISTYPE 'com';  "RAS line
CAS    pin   19  ISTYPE 'com';  "CAS line
!DRAM_WE pin  20  ISTYPE 'com';  "Write Enable Line to DRAM
!R_A_EN pin   21 ISTYPE 'com';  "Buffer row address enable
!C_A_EN pin   22 ISTYPE 'com';  "Buffer column address enable line
"      pin   23                  input   unused
"      pin   24                  input   unused
!NCS   pin   25;                "input chip select
!NWE   pin   26;                "input for whether write access
!NWAIT pin   27  ISTYPE 'reg';  "output for making cpu wait for refresh finish
!MCLK   pin   28;                "main clock input
!RCLK   pin   29;                "refresh clock input
"      pin   13                  output  unused
"      pin   14                  output  unused
"      pin   15                  output  unused
"      pin   16                  output  unused
St0    pin 37      ISTYPE 'reg';  "output  state bit 0
St1    pin 38      ISTYPE 'reg';  "output  state bit 1
St2    pin 39      ISTYPE 'reg';  "output  state bit 2
St3    pin 40      ISTYPE 'reg';  "output  state bit 3
St4    pin 41      ISTYPE 'reg';  "output  state bit 4
RefRqst pin      ISTYPE 'reg';  "refresh request bit
RefInProgress pin   ISTYPE  'reg_SR';   "bit indicating refresh just finished, used to
implement logic for holding nwait one more cycle
RefInProgressDelay pin  ISTYPE  'reg_D';
"      pin   17                  output  unused
"      pin   18                  output  unused



"the states
```

```
StateBits   =  [ St4, St3, St2, St1, St0 ];     " state bits
                                              " state assignments
Idle        =  [   0,   0,   0,   0,   0 ];     " idle state (waiting for a cycle to start)
Access0     =  [   0,   0,   0,   0,   1 ];     " enable row address for Access
Access1     =  [   0,   0,   0,   1,   0 ];     " RAS line low
Access2     =  [   0,   0,   0,   1,   1 ];     " disable row address for Access
Access3     =  [   0,   0,   1,   0,   0 ];     " enable column address for Access
Access4     =  [   0,   0,   1,   0,   1 ];     " CAS line low
Access5     =  [   0,   0,   1,   1,   0 ];     " Wait for data access
Access6     =  [   0,   0,   1,   1,   1 ];     " Wait for data access
Access7     =  [   0,   1,   0,   0,   0 ];     " Wait for data access
Access8     =  [   0,   1,   0,   0,   1 ];     " RAS, CAS line high
Access9     =  [   0,   1,   0,   1,   0 ];     " Pre-charge
Access10    =  [   0,   1,   0,   1,   1 ];     " Pre-charge
Ref0        =  [   0,   1,   1,   0,   0 ];     " CAS low
Ref1        =  [   0,   1,   1,   0,   1 ];     " RAS low
Ref2        =  [   0,   1,   1,   1,   0 ];     " Wait
Ref3        =  [   0,   1,   1,   1,   1 ];     " Wait
Ref4        =  [   1,   0,   0,   0,   0 ];     " Wait
Ref5        =  [   1,   0,   0,   0,   1 ];     " RAS, CAS high
Ref6        =  [   1,   0,   0,   1,   0 ];     " Pre-charge
Ref7        =  [   1,   0,   0,   1,   1 ];     " Pre-charge
Inval0      =  [   1,   0,   1,   0,   0 ];
Inval1      =  [   1,   0,   1,   0,   1 ];
Inval2      =  [   1,   0,   1,   1,   0 ];
Inval3      =  [   1,   0,   1,   1,   1 ];
Inval4      =  [   1,   1,   0,   0,   0 ];
Inval5      =  [   1,   1,   0,   0,   1 ];
Inval6      =  [   1,   1,   0,   1,   0 ];
Inval7      =  [   1,   1,   0,   1,   1 ];     " Invalid state
Inval8      =  [   1,   1,   1,   0,   0 ];     " Invalid state
Inval9      =  [   1,   1,   1,   0,   1 ];     " Invalid state
Inval10     =  [   1,   1,   1,   1,   0 ];     " Invalid state
Inval11     =  [   1,   1,   1,   1,   1 ];     " Invalid state


EQUATIONS


" NWAIT equation
NWAIT := (NCS & (RefInProgress));


NTRST = NReset & JTAG_NTRST;
DRAM_WE = NWE & NCS;


" clocks for the registered outputs (state bits)
StateBits.CLK  =  MCLK;        " use the global clock pin
NWAIT.CLK = !MCLK;

RefInProgress.S = (StateBits == Ref0);
RefInProgress.R = (StateBits == Idle);
RefInProgress.CLK = MCLK;
RefInProgressDelay.D = RefInProgress;
RefInProgressDelay.CLK = MCLK;
RefRqst.CLK = RCLK;
RefRqst.AR = (StateBits == Ref7);
RefRqst := 1;

STATE_DIAGRAM  StateBits        " a Mealy state machine


STATE  Idle:                    " in the idle state waiting for an access
```

```
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN Idle;
   ELSE IF (RefRqst) THEN  Ref0;
   "ELSE IF (NCS & NWE) THEN Write0;
   ELSE IF (NCS) THEN      Access0;
   ELSE                       Idle;     " otherwise just stay here

STATE Access0:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 1;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access1;

STATE Access1:

   RAS = 0;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 1;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access2;

STATE Access2:

   RAS = 0;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access3;

STATE Access3:

   RAS = 0;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access4;

STATE Access4:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access5;
```

```
STATE Access5:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access6;

STATE Access6:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access7;

STATE Access7:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 1;

   IF (NReset) THEN        Idle;
   ELSE                   Access8;


STATE Access8:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access9;

STATE Access9:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                   Access10;


STATE Access10:

   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;
```

```
   GOTO Idle;

STATE Ref0:

   RAS = 1;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                    Ref1;

STATE Ref1:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                    Ref2;


STATE Ref2:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   IF (NReset) THEN        Idle;
   ELSE                    Ref3;

STATE Ref3:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                    Ref4;


STATE Ref4:

   RAS = 0;
   CAS = 0;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                    Ref5;


STATE Ref5:


   RAS = 1;
```

```
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                    Ref6;

STATE Ref6:


   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;


   IF (NReset) THEN        Idle;
   ELSE                    Ref7;

STATE Ref7:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval0:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval1:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval2:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval3:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;
```

```
STATE Inval4:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval5:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval6:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval7:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval8:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval9:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval10:
   RAS = 1;
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;

STATE Inval11:
   RAS = 1;
```

```
   CAS = 1;
   "DRAM_WE = 1;
   R_A_EN = 0;
   C_A_EN = 0;

   GOTO                    Idle;




TEST_VECTORS

( [  MCLK,  RCLK, NReset,  NCS,  NWE ] -> [   RAS, CAS, DRAM_WE, R_A_EN, C_A_EN, NWAIT,
RefRqst, St0, St1, St2, St3, St4 ] )

  [     0,     0,      0,    0,   0  ] -> [   .X., .X.,     .X.,    .X.,    .X.,   .X.,
  .X., .X., .X.,  .X., .X., .X. ];


" reset the system
  [   .C.,     0,      1,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  .X., .X., .X.,  .X., .X., .X. ];
  [   .C.,     0,      1,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  .X., .X., .X.,  .X., .X., .X. ];
  [   .C.,     0,      1,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  .X., .X., .X.,  .X., .X., .X. ];

" Single Access
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access0
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access1
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access2
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access3
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access4
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access5
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle

" Single write
  [   .C.,     0,      0,    1,   1  ] -> [     1,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write0
  [   .C.,     0,      0,    1,   1  ] -> [     0,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write1
  [   .C.,     0,      0,    1,   1  ] -> [     0,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write2
  [   .C.,     0,      0,    1,   1  ] -> [     0,   1,       0,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write3
  [   .C.,     0,      0,    1,   1  ] -> [     0,   0,       0,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write4
  [   .C.,     0,      0,    1,   1  ] -> [     0,   0,       0,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write5
  [   .C.,     0,      0,    1,   1  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Write6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
```

```
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle

" Consecutive Accesss
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access0
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access1
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access2
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access3
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access4
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access5
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access0
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      1,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access1
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access2
  [   .C.,     0,      0,    1,   0  ] -> [     0,   1,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access3
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access4
  [   .C.,     0,      0,    1,   0  ] -> [     0,   0,       1,      0,      1,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access5
  [   .C.,     0,      0,    1,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Access6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Pre-charge
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle

" Refresh
  [     0,   .C.,      0,    0,   0  ] -> [   .X., .X.,     .X.,    .X.,    .X.,   .X.,
  1, .X., .X.,  .X., .X., .X. ]; "Trigger the refresh request
  [   .C.,     0,      0,    0,   0  ] -> [     1,   0,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref0
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref1
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref2
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref3
  [   .C.,     0,      0,    0,   0  ] -> [     0,   0,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref4
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref5
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  1, .X., .X.,  .X., .X., .X. ]; "Ref6
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0, .X., .X.,  .X., .X., .X. ]; "Ref7
```

```
  [   .C.,     0,      0,    0,   0  ] -> [     1,   1,       1,      0,      0,     0,
  0,   0,   0,    0,   0,   0 ]; "Idle


END DRAM
```

Appendix E (Software)

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                   Boot.inc
@                       Contains definitions used by boot.s
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017

@@@@@ SMC Definitions


    @ SRAM
    @ RWHOLD,RWSETUP    0x00------      1/2 cycle setup and hold
    @ ACSS              0x--00----      Standard chip select waveform
    @                   0x----30--      connected to 16-bit device
    @ 1 Wait state      0x------81      Enable 1 wait state
    .equ SMC_CSR1_VAL,  0x00003081

    @ DRAM
    @ RWHOLD,RWSETUP    0x30------      3 RWHOLD
    @ ACSS              0x--00----      Standard chip select waveform
    @ 8 bit             0x----43--      Connected to 8-bit device with 3 clock cycles data float
    @ 7 Wait state      0x------87      Enable 7 wait state
    .equ SMC_CSR2_VAL,  0x30004387

    @ ROM
    @ RWHOLD,RWSETUP    0x00------      1/2 cycle setup and hold
    @ ACSS              0x--00----      Standard chip select waveform
    @                   0x----40--      Connected to 8-bit device
    @ 3 Wait state      0x------83      Enable 3 wait state
    .equ SMC_CSR7_VAL,  0x00004083



@@@@@ PMC Definitions


    .equ PMC_PLLAR_VAL, 0x20153F05  @ MULA = 21 + 1, DIVA = 5, PLLACOUNT =63, OUTA = 00
    .equ PMC_MCKR_VAL,  0x00000006  @0x0000000E for dram
    .equ PMC_PCK0_VAL,  0x00000006
    .equ PMC_PCER_VAL,  0x0000001C  @Enable PIOA, PIOB
    .equ PMC_PCK1_VAL,  0x00000019  @MCK source, Pre-scale by 64

    .equ PMC_MCKR_MDIV, 0           @Run cpu 1+1 times faster than MCK

@@@@@ PIOA Definitions

    .equ PIOA_BSR_VAL,  0x01000000
    .equ PIOA_OER_VAL,  0x01000000
    .equ PIOA_PDR_VAL,  0x01000000

@@@@@ PIOB Definitions

    .equ PIOB_ASR_VAL,  0x08000000
    .equ PIOB_OER_VAL,  0x08000000
    .equ PIOB_PDR_VAL,  0x08000000
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                                                                                @
@ boot.s                                                                         @
@                                                                                @
@ Bootloader for the EE52 ARM VoIP phone project.  It sets up any registers     @
@ needed to copy the main program from the external parallel ROM to the         @
@ external SRAM.  This includes the following peripherals:                      @
@                                                                                @
@   - Clock                                                                      @
@   - ROM and SRAM chips selects                                                 @
@                                                                                @
@ It then copies the main program from the ROM to the SRAM and jumps to the     @
@ beginning of code in the SRAM.                                                 @
@                                                                                @
@ Revision History:                                                              @
@                                                                                @
@   2010/02/02  Joseph Schmitz  Modified code from Arthur Chang to make it      @
@                               available to the students.                      @
@                                                                                @
@   2011/01/27  Joseph Schmitz  Split from crt0.s to boot.s                      @
@                                                                                @
@   2011/01/31  Joseph Schmitz  Updated exception vectors to include special    @
@                               word value at ARM vector 6. (see 13.3.2)        @
@                                                                                @
@   2011/02/06  Joseph Schmitz  Added documentation on exception vectors.       @
@                                                                                @
@   2017/05/21  William Werst   Modified to work for this project               @
@                                                                                @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

.include    "at91rm9200.inc"
.include    "system.inc"
.include    "macro.inc"
.include    "boot.inc"


.text
.arm


@@@ Exception Vectors @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@ Start of the IRQ vector table.  This defines the interrupt handler for each
@ type of interrupt.  Must be at the memory address 0x0 (remapped at boot).
@
@ Exception                   Description
@
@ Reset                       Occurs when the processor reset pin is asserted.
@                             This exception is only expected to occur for
@                             signalling power-up, or for resetting as if the
@                             processor has just powered up. A soft reset can be
@                             done by branching to the reset vector (0x0000).
@
@ Undefined Instruction       Occurs if neither the processor, or any attached
@                             coprocessor, recognizes the currently executing
@                             instruction.  Software Interrupt (SWI) This is a
@                             user-defined synchronous interrupt instruction. It
@                             allows a program running in User mode, for example,
@                             to request privileged operations that run in
@                             Supervisor mode, such as an RTOS function.
@
@ Sofwarte Interrupt          Occurs when the processor generates a software
@                             interrupt.
@
@ Prefetch Abort              Occurs when the processor attempts to execute an
@                             instruction that has prefetched from an illegal
```

```
@                           address.
@
@ Data Abort                Occurs when a data transfer instruction attempts to
@                           load or store data at an illegal address.
@
@ Vector 6                  Used to specify the number of bytes to download from
@                           and external boot memory into internal SRAM on
@                           reset.
@
@ IRQ                       Occurs when the processor external interrupt request
@                           pin is asserted (LOW) and the I bit in the CPSR is
@                           clear.
@
@ FIQ                       Occurs when the processor external fast interrupt
@                           request pin is asserted (LOW) and the F bit in the
@                           CPSR is clear.
@
@
@ The format is as follows:
@
@   reset:    Reset vector
@   undef:    Undefine instruction
@   swi:      Software interrupt
@   abort:    Prefect abort
@   data:     Data abort
@   btldr:    Defines how much data to download from the boot memory
@   irq:      Normal interrupt (low priority).  Actual ISR address loaded from AIC
@   fiq:      Fast interrupt (high priority)

IRQTable:
.org 0x0

    reset:
        B   _start
    undef:
        B   undef
    swi:
        B   swi
    prefetch:
        B   prefetch
    data:
        B   data
    btldr:
        .word 0x00000008
    irq:
        LDR PC, [PC, #-0xF20]
    fiq:
        B   fiq



@@@ Startup Code (system reset restarts here) @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
.org 0x20

.global _start
_start:

@ In this file you must do the following (at least for now):
@
@   - Switch to the master clock
@
@   - Wait for it to stabilize.  The datasheet tells you how long this will
@     take.  You will need to force your CPU to do no external memory
@     operations during this time.  There is an easy way to do this, but
@     think about it and only ask me if you still can't come up with anything.
@
```

```
@    - Set up the chip selects for SRAM/ROM
@


@@@ Clock Initialization @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @Configure PLLA for DIVA = 5, MULA = 22

    mSET_HREG PMC_PLLAR, PMC_PLLAR_VAL

    @Configure MCK @MDIV = 00, PRES = 001, CSS = 10
    @First, need to change one value at a time, so change CSS first
    LDR r0,         =PMC_MCKR           @Load PMC_MCKR address
    LDR r1,         [r0]                @Load current PMC_MCKR value
    LDR r0,         =PMC_MCKR_VAL       @Check current value to see if any changes
    CMP r1,         r0                  @Because clock ready bit will not go
    BEQ DoneMCK                         @high if the same value is re-written.

    AND r1,     r1, #0xFFFFFFFC         @Mask out CSS bits
    LDR r0, =(PMC_MCKR_VAL & 0x03)      @Load the CSS component of final MCKR value
    ORR r1,         r0                  @Merge current PRES value with new CSS
    LDR r0,         =PMC_MCKR           @Load PMC_MCKR address
    STR r1, [r0]                        @Store intermediate value in MCKR
    LDR r2, =2000                       @Clock initialization timeout
WaitMCKRDY:
    SUBS r2, r2, #1                     @Timeout
    BLO DoneMCKRDY
    LDR r0, =PMC_SR
    LDR r1, [r0]
    TST r1, #0x8
    BEQ WaitMCKRDY
    @B DoneMCKRDY
DoneMCKRDY:


    mSET_HREG PMC_MCKR, PMC_MCKR_VAL
    mSET_HREG PMC_MCKR, (PMC_MCKR_VAL | (PMC_MCKR_MDIV << 8))
    @B DoneMCK
DoneMCK:
    @Configure Peripheral clock
    mSET_HREG PMC_PCER, PMC_PCER_VAL

    @Configure PCK0
    mSET_HREG PMC_PCK0, PMC_PCK0_VAL

    @Configure PCK1
    @mSET_HREG PMC_PCK1, PMC_PCK1_VAL

    @Configure PIOA
    @mSET_HREG PIOA_BSR, PIOA_BSR_VAL
    @mSET_HREG PIOA_OER, PIOA_OER_VAL
    @mSET_HREG PIOA_PDR, PIOA_PDR_VAL

    @Configure PIOB
    mSET_HREG PIOB_ASR, PIOB_ASR_VAL
    mSET_HREG PIOB_OER, PIOB_OER_VAL
    mSET_HREG PIOB_PDR, PIOB_PDR_VAL



    @Enable PCK0, PCK1 output
    mSET_HREG 0xFFFFFC00, 0x00000300


    @@@ CS Initialization @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

    @Setup SRAM
```

```
    mSET_HREG SMC_CSR1, SMC_CSR1_VAL

    @Setup DRAM
    mSET_HREG SMC_CSR2, SMC_CSR2_VAL

    @Setup ROM
    mSET_HREG SMC_CSR7, SMC_CSR7_VAL

    @@@ Verify DRAM and SRAM are functioning
    @@Check SRAM valid

    @Test SRAM
    LDR r0, =SRAM_START
    LDR r1, =SRAM_SIZE
    BL mem_test
    CMP     r0,     #TRUE
    BNE     memtestfail

    @Test DRAM
    LDR r0, =DRAM_START
    LDR r1, =DRAM_SIZE
    BL mem_test
    CMP     r0,     #TRUE
    BNE     memtestfail

    @@@ Copy Code from External ROM -> External SRAM @@@@@@@@@@@@
    LDR     r0,     =SRAM_SIZE - 4
    LDR     r1,     =ROM_START
    LDR     r2,     =SRAM_START
CopyROMToSRAM:
    LDR     r3,     [r1, r0]
    STR     r3,     [r2, r0]
    SUBS    r0,     #4
    BHS     CopyROMToSRAM

    @Check that code loaded into SRAM matches code in ROM
    LDR     r0,     =SRAM_SIZE - 4
    LDR     r1,     =ROM_START
    LDR     r2,     =SRAM_START
CheckCopyToSRAM:
    LDR     r3,     [r1, r0]
    LDR     r4,     [r2, r0]
    CMP     r3,     r4
    BNE     LoadToSRAMFailed
    SUBS    r0,     #4
    BHS     CheckCopyToSRAM
    @@@ Branch to the Main Body of Code Now Located in the External SRAM @@@@@@@@@@@@

    @Uncomment this to branch to the copied code
    BL      SRAM_START

@If don't want to branch to low_level_init, fall through to BootEndLoop
BootEndLoop:
    B BootEndLoop

memtestfail:
    B memtestfail

LoadToSRAMFailed:
    B LoadToSRAMFailed

.end
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                                                                              @
@ crt0.s                                                                       @
@                                                                              @
@ Initialization file for EE52 ARM VoIP phone project.  It sets up the IRQ     @
@ vector table, initializes the stacks for both the IRQ and System modes,      @
@ sets up the Master Clock, all of the chip selects for external memories,     @
@ and will eventually call all of the intialization functions for each         @
@ hardware block.  Finally it invokes the main user interface code.            @
@                                                                              @
@                                                                              @
@ Revision History:                                                            @
@                                                                              @
@   2008/02/02  Joseph Schmitz  Modified code from Arthur Chang to make it     @
@                               available to the students.                     @
@   2011/01/27  Joseph Schmitz  Split into crt0.s and boot.s.                  @
@   2011/01/31  Joseph Schmitz  Removed unused comments.                       @
@   2012/01/24  Glen George     Updated comments, modified included files,     @
@                               and cleaned up the code a little.              @
@   2012/01/29  Glen George     Added comments explaining initialization.      @
@                                                                              @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

.include    "at91rm9200.inc"
.include    "system.inc"
.include    "macro.inc"
.include    "interfac.inc"

.text
.arm

@ In this file you must do at least the following:
@
@   - As you write other functions/code for the various hardware blocks, you
@     will call the initialization functions for them from here as well.



@ low_level_init
@
@ Description: Initializes the ARM processor modes, calls the
@              initialization functions for the different parts
@              of the code, and then calls the main loop.
@
@ Operational Description: Initializes arm processor modes,
@                          then goes through and calls a sequence
@                          of initialization functions for various
@                          files. Finally, the main loop is called
@                          (or for the audio demo, audio_demo is called)
@
@ Arguments: None
@
@ Return values: never returns
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
```

```
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global low_level_init
low_level_init:

    @Uncomment instruction below to prevent ROM code from executing after
    @bootloader runs. This effectively creates a dummy ROM code that does nothing.
    @B low_level_init

@ Stack and IRQ Initialization

    LDR     r0, =TOP_STACK                  @load address for top of the stack

    @ put the CPU in interrupt mode and set the stack pointer for this mode
    MSR     cpsr_c, #ARM_MODE_IRQ | I_BIT | F_BIT
    MOV     sp, r0

    @ adjust the starting stack address for the interrupt stack just setup
    SUB     r0, r0, #IRQ_STACK_SIZE

    @ put the CPU in supervisor mode and set the stack pointer for this mode
    MSR     cpsr_c, #ARM_MODE_SVC | I_BIT | F_BIT
    MOV     sp, r0

    @ adjust the starting stack address for the supervisor mode stack just setup
    SUB     r0, r0, #SVC_STACK_SIZE

    @ finally, put the CPU in user mode and set the stack pointer for this mode
    MSR     cpsr_c, #ARM_MODE_USR | F_BIT
    MOV     sp, r0



@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@  user initialization goes here  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    BL init_system                          @ Initialize the system
    BL keypad_init                          @ Initialize the keypad
    BL audio_init                           @ Initialize the audio
    BL display_init                         @ Initialize the display

loop:
    @BL      audioDemo              @ Uncomment this line to run the audioDemo
    BL      main                    @ run the main function (no arguments)

    B       low_level_init          @ if main returns (shouldn't)
                                    @ reinitialize everything and start
                                    @ over

.end
```

```
@ Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                    Audio.inc
@                        Contains definitions used by audio.s
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ Description: Defines register values used by the audio system.
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017

    @ 0000 0000 0000 0000 ---- ---- ---- ----       Reserved
    @ ---- ---- ---- ---- 1--- ---- ---- ----       Software reset
    @ ---- ---- ---- ---- -000 00-- ---- ----       Reserved
    @ ---- ---- ---- ---- ---- --01 ---- ----       Enable transmit
    @ ---- ---- ---- ---- ---- ---- 0000 00--       Reserved
    @ ---- ---- ---- ---- ---- ---- ---- --01       Enable receive
    .equ    SSC0_CR_VAL, 0x00008101

    .equ    SSC0_CMR_VAL, 0x0000000B                @CLK = 48 MHz / (2*(10+1) = 2 MHz
    @ 0111 1111 ---- ---- ---- ---- ---- ----       Period: 2*(127+1) = 256 bits
    @ ---- ---- 0000 0000 ---- ---- ---- ----       STTDLY: No delay
    @ ---- ---- ---- ---- 0000 ---- ---- ----       Reserved
    @ ---- ---- ---- ---- ---- 0001 ---- ----       START: Transmit Start
    @ ---- ---- ---- ---- ---- ---- 00-- ----       Reserved
    @ ---- ---- ---- ---- ---- ---- --0- ----       CKI: Rising edge
    @ ---- ---- ---- ---- ---- ---- ---0 00--       CKO: No clock output
    @ ---- ---- ---- ---- ---- ---- ---- --01       CKS: TK Clock signal
    @ ---- ---- ---- ---- ---- ---- ---- ----       Reserved
    @ 0111 1111 0000 0001 0000 0001 0000 0001
    .equ    SSC0_RCMR_VAL, 0x7F000101           @Period 256 bits, 0 start delay, start on
    transmit, Clock on falling edge, TK Clock signal is clock


    @ 0000 000- ---- ---- ---- ---- ---- ----       Reserved
    @ ---- ---0 ---- ---- ---- ---- ---- ----       FSEDGE: Positive Edge
    @ ---- ---- 0--- ---- ---- ---- ---- ----       Reserved
    @ ---- ---- -000 ---- ---- ---- ---- ----       FSOS: None
    @ ---- ---- ---- 0000 ---- ---- ---- ----       FSLEN: 0
    @ ---- ---- ---- ---- 0000 ---- ---- ----       Reserved
    @ ---- ---- ---- ---- ---- 0000 ---- ----       DATNB: 0+1
    @ ---- ---- ---- ---- ---- ---- 1--- ----       MSBF: Most significant Bit first
    @ ---- ---- ---- ---- ---- ---- -0-- ----       Reserved
    @ ---- ---- ---- ---- ---- ---- --0- ----       LOOP: False
    @ ---- ---- ---- ---- ---- ---- ---0 1111       DATLEN: 16 bits per bit stream
    @ 0000 0000 0010 0000 0000 0000 1000 1111
    .equ    SSC0_RFMR_VAL, 0x0020008F           @Positive edge frame sync, positive pulse 1
    clock, 16 DATNB

    @ 0111 1111 ---- ---- ---- ---- ---- ----       Period: 2*(127+1) = 256 bits
    @ ---- ---- 0000 0000 ---- ---- ---- ----       STTDLY: No delay
    @ ---- ---- ---- ---- 0000 ---- ---- ----       Reserved
    @ ---- ---- ---- ---- ---- 0011 ---- ----       START: High TF Signal
    @ ---- ---- ---- ---- ---- ---- 00-- ----       Reserved
    @ ---- ---- ---- ---- ---- ---- --1- ----       CKI: Rising edge
    @ ---- ---- ---- ---- ---- ---- ---0 01--       CKO: Continuous Transmit Clock
    @ ---- ---- ---- ---- ---- ---- ---- --00       CKS: Divided clock
    @ ---- ---- ---- ---- ---- ---- ---- ----       Reserved
```

```
    @ 0111 1111 0000 0000 0000 0011 0020 0100
    .equ    SSC0_TCMR_VAL, 0x7F000324

    @ 0000 000- ---- ---- ---- ---- ---- ----       Reserved
    @ ---- ---0 ---- ---- ---- ---- ---- ----       FSEDGE: Positive Edge
    @ ---- ---- 1--- ---- ---- ---- ---- ----       FSDEN: Shift during transmit frame sync
    signal
    @ ---- ---- -010 ---- ---- ---- ---- ----       FSOS: Positive pulse
    @ ---- ---- ---- 0000 ---- ---- ---- ----       FSLEN: 0
    @ ---- ---- ---- ---- 0000 ---- ---- ----       Reserved
    @ ---- ---- ---- ---- ---- 0000 ---- ----       DATNB: 0+1
    @ ---- ---- ---- ---- ---- ---- 1--- ----       MSBF: Most significant Bit first
    @ ---- ---- ---- ---- ---- ---- -0-- ----       Reserved
    @ ---- ---- ---- ---- ---- ---- --0- ----       DATDEF: 0 is default bit
    @ ---- ---- ---- ---- ---- ---- ---0 1111       DATLEN: 16 bits per bit stream
    @ 0000 0000 1010 0000 0000 0000 0000 1111
    .equ    SSC0_TFMR_VAL, 0x0020008F



    .equ    AUDIO_VOLUME,   0x07                    @Define system volume
```

```
@done
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@ Audio.s
@
@ Description: Contains code for controlling the audio component of
@ the EE52 VoIP Project
@
@ Table of Contents:
@   - audio_init: Call to initialize this code file before
@                   calling anything else
@   - call_start: Call to initiate a call
@   - call_halt:  Call to halt a call that has been started with call_start
@   - update_rx:  Handler used to update the receive buffer
@   - update_tx:  Handler used to update the transmit buffer
@   - setVolume:  Call to set the volume
@   - audioDemo:  A test function used to demo the audio code on its own,
@                 by looping the audio input back to the audio output.
@
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

.include    "at91rm9200.inc"
.include    "system.inc"
.include    "interfac.inc"
.include    "audio.inc"
.include    "macro.inc"


.text
.arm



@ audio_init
@
@ Description: Call to initialize everything in this file
@
@ Operational Description: Initializes the registers to the
@                          initialization values for this project
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: SSC0 (audio serial communication) - initialized
@
@ Outputs: SSC0 (audio serial communication) - initialized
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
```

```
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global audio_init
audio_init:
    mSTARTFNC
    mSET_HREG   PMC_PCER,   (1 << 14)       @Enable the peripheral clock
    mSET_HREG   SSC0_CR,    SSC0_CR_VAL     @Setup the serial controller
    mSET_HREG   SSC0_CMR,   SSC0_CMR_VAL    @Setup the serial clock rate
    mSET_HREG   SSC0_RCMR,  SSC0_RCMR_VAL   @Setup receive clock mode
    mSET_HREG   SSC0_RFMR,  SSC0_RFMR_VAL   @Setup receive frame mode
    mSET_HREG   SSC0_TCMR,  SSC0_TCMR_VAL   @Setup transmit clock mode
    mSET_HREG   SSC0_TFMR,  SSC0_TFMR_VAL   @Setup transmit frame mode
    mSET_HREG   PIOB_ASR,   0xF             @Setup serial output pins
    mSET_HREG   PIOB_PDR,   0xF             @Setup serial output pins
    mSET_HREG   PIOB_OER,   0xF             @Setup serial output pin
    mSET_HREG   SSC0_CR,    0x00000101      @Enable serial
    mSET_HREG   SSC0_PTCR,  0x00000101      @Enable DMA
    mRETURNFNC


@ call_start
@
@ Description: Initializes a call with the initial receive buffer pointer (r0)
@
@ Operational Description: Passes the initial buffer pointer (r0) to the
@                          function that handles adding new buffers
@                          to the system.
@
@ Arguments: r0 - pointer to first buffer in DRAM
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: update_rx should be called, not update_tx. This ends up being
@             fine for long term operation because only the first buffer
@             played is affected, which is why it worked in the demo.
```

```
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global call_start
call_start:
    mSTARTFNC                               @Call start function macro
    BL  update_tx
    mRETURNFNC                              @Call return from function macro


@ call_halt
@
@ Description: Halts the current call regardless of the state of buffers.
@              If the call is already halted, this has no effect.
@
@ Operational Description: The counter registers for the DMA engine are
@                          cleared. This will stop any memory accesses
@                          from occurring through the DMA and into the
@                          buffers.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: SSC0 - the counter registers for DMA are cleared
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global call_halt
call_halt:
    mSTARTFNC                               @Call start function macro
    @Clear the counters in all DMA registers for audio
    mSET_HREG   SSC0_RNCR, 0                @Clear the counter register for the
                                            @next DMA buffer receive
    mSET_HREG   SSC0_RCR, 0                 @Clear the counter register for the
                                            @buffer currently being received
    mSET_HREG   SSC0_TNCR, 0                @Clear the counter register for the
                                            @next DMA buffer transmit
```

```
    mSET_HREG   SSC0_TCR, 0                 @Clear the counter register for the
                                            @buffer currently being transmitted
    mRETURNFNC                              @Call return from function macro


@ update_rx
@
@ Description: Takes a pointer, and either uses it and returns true, or
@              discards it and returns false.
@
@ Operational Description: The next counter register is pulled from
@                          the DMA engine and compared to 0, to see
@                          if the next register in DMA is empty. If
@                          it is, the new register is added to the
@                          DMA engine, and true is returned. Else,
@                          false is returned.
@
@ Arguments: r0 - pointer to new record buffer
@
@ Return values: bool - true if passed buffer is used, else false.
@
@ Local variables:
@
@ Shared variables:
@
@ Global Variables:
@
@ Inputs: SSC0 - sets up DMA receive
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global update_rx
update_rx:
    mSTARTFNC                           @Call start function macro
    mLOADTOREG  r1, SSC0_RNCR           @Load the length of the next receive
                                        @buffer currently queued up
    CMP     r1,     #0                  @Check if next receive buffer is empty
    LDRNE   r1,     =FALSE              @If memory is not empty
                                        @Post-demo comment: I'm pretty sure this
                                        @should have been r0, not sure why this
                                        @worked
    BNE     endUpdate_RX                @return false since new buffer not needed
    @BEQ    rx_empty                    @Call return function macro
rx_empty:
    mSTOREFROMREG   r0, r1, SSC0_RNPR   @Store r0 (next pointer) into next
                                        @pointer register
    mSET_HREG   SSC0_RNCR,  (AUDIO_BUFLEN /2) - 1 @Store buffer length in half-words
    LDR     r0,     =TRUE               @Return success
```

```
endUpdate_RX:
    mRETURNFNC


@ update_tx
@
@ Description: Takes a transmit buffer pointer, and either uses it and returns
@              true, or discards it and returns false.
@
@ Operational Description: The next counter register is pulled from
@                          the DMA engine and compared to 0, to see
@                          if the next register in DMA is empty. If
@                          it is, the new register is added to the
@                          DMA engine, and true is returned. Else,
@                          false is returned.
@
@ Arguments: r0 - pointer to new transmit buffer
@
@ Return values: bool - true if passed buffer is used, else false.
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: SSC0 - sets up DMA transmit
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global update_tx
update_tx:
    mSTARTFNC
    mLOADTOREG  r1, SSC0_TNCR
    CMP     r1,     #0                  @Check if next transmit buffer is empty
    LDRNE   r1,     =FALSE              @If memory is not empty
                                        @Post-demo comment: I'm pretty sure this
                                        @should have been r0, not sure why this
                                        @worked
    BNE     endUpdate_TX                @return false since new buffer not needed
    @BEQ    tx_empty
tx_empty:
    PUSH    {r0-r3}                     @Store registers to free up for temp
    LDR     r1, =(AUDIO_BUFLEN -2)      @Load buffer length
    LDR     r2, =AUDIO_VOLUME           @Load the volume setpoint
    BL      setVolume                   @Set volume
    POP     {r0-r3}                     @Restore register
    mSTOREFROMREG   r0, r1, SSC0_TNPR   @Save next transmit register
```

```
    mSET_HREG   SSC0_TNCR,  (AUDIO_BUFLEN / 2) - 1 @Save length of next transmit
                                        @Register in half-words
    LDR     r0,     =TRUE               @Return true
endUpdate_TX:
    mRETURNFNC


@ setVolume
@
@ Description: Goes through the transmit buffer and sets the volume bits
@
@ Operational Description: The transmit buffer is looped over in reverse, and
@                          each byte is OR-masked with the volume, and saved back
@                          in-place in the buffer.
@
@
@ Arguments: r0 - pointer to transmit buffer to set volume for
@            r1 - length of transmit buffer in bytes
@            r2 - volume, encoded as the bits to preserve in each byte
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

setVolume:
    mSTARTFNC
updateValue:
    LDRH r3, [r0,r1]                @Get byte of audio
    ORR  r3, r3, r2                 @Set the high bits for volume control
    STRH r3, [r0,r1]                @Store byte of audio
    SUB r1, #2                      @Decrement to next byte
    CMP r1, #0                      @Check if at last byte
    BGE updateValue                 @If not, continue
    mRETURNFNC                      @Return


@ audioDemo
@
@ Description: Audio is looped back from the microphone to the speaker
@              continuously
@
```

```
@ Operational Description: Five buffers are looped through, starting with
@                          receive as buffer 1 and transmit as buffer 3.
@                          The loop is unrolled, and the loop has no exit
@                          condition.
@
@ Arguments: None
@
@ Return values: None (never returns)
@
@ Local variables: Buf[1..5] - buffers for storing audio data
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: Circular buffer
@
@ Limitations: Loop is unrolled, and the buffers are defined separately
@              rather than as an array. Changing buffer count is not
@              trivial.
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global audioDemo
audioDemo:
    mSTARTFNC
loopAudioDemo:
    Buf1_rx:
    LDR     r0, =Buf1
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf1_rx

    LDR     r0, =Buf3
    BL  update_tx

Buf2_rx:
    LDR     r0, =Buf2
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf2_rx

    LDR     r0, =Buf4
    BL  update_tx

Buf3_rx:
    LDR     r0, =Buf3
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf3_rx
```

```
    LDR     r0, =Buf5
    BL  update_tx

Buf4_rx:
    LDR     r0, =Buf4
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf4_rx

    LDR     r0, =Buf1
    BL  update_tx

Buf5_rx:
    LDR     r0, =Buf5
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf5_rx

    LDR     r0, =Buf2
    BL  update_tx

    B   loopAudioDemo
    mRETURNFNC

.data

.balign 4
Buf1:           @Audio buffer 1
    .skip 256
Buf2:           @Audio buffer 2
    .skip 256
Buf3:           @Audio buffer 3
    .skip 256
Buf4:           @Audio buffer 4
    .skip 256
Buf5:           @Audio buffer 5
    .skip 256

.end
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                    display.inc
@                        Contains definitions used by display.s
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ Description: Defines register values used by the NHDC12832 Display driver code
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017

@Pin definitions
    .equ    DISP_MOSI,  1                       @MOSI pin is A1
    .equ    DISP_SPCK,  2                       @Clock pin is A2
    .equ    DISP_CS,    3                       @Chip Select pin is A3
    .equ    DISP_A0,    4                       @Mode select pin is A4
    .equ    DISP_BCKLIGHT, 5                    @Backlight pin is A5
    .equ    DISP_RST,   6                       @Reset pins is A6

@SPI Status register bit definitions
    .equ    SPI_SR_TXBUFE,  0x80

@SPI Handler definitions
    .equ    COM_QUEUE_LENGTH,   16

@Register definitions

    @ 0000 0000 0000 0000 0000 0000 ---- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- 1--- ---- SWRST
    @ ---- ---- ---- ---- ---- ---- -000 00--
    @ ---- ---- ---- ---- ---- ---- ---- --10 Disable
    @ 0000 0000 0000 0000 0000 0000 1000 0010
    .equ    SPI_CR_RESET, 0x00000082

    @ 0000 0000 0000 0000 0000 0000 ---- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- 0--- ---- SWRST
    @ ---- ---- ---- ---- ---- ---- -000 00--
    @ ---- ---- ---- ---- ---- ---- ---- --01 Enable
    @ 0000 0000 0000 0000 0000 0000 0000 0001
    .equ    SPI_CR_EN, 0x00000001

    @ 0000 0000 0000 0000 0000 0000 ---- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- 0--- ---- SWRST
    @ ---- ---- ---- ---- ---- ---- -000 00--
    @ ---- ---- ---- ---- ---- ---- ---- --10 Disable
    @ 0000 0000 0000 0000 0000 0000 0000 0010
    .equ    SPI_CR_DIS, 0x00000002

    @ 0000 0000 ---- ---- ---- ---- ---- ---- DLYBCS: 0
    @ ---- ---- 0000 ---- ---- ---- ---- ---- Reserved
    @ ---- ---- ---- 1110 ---- ---- ---- ---- PCS: NPCS0 is where display is
    @ ---- ---- ---- ---- 0000 0000 ---- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- 0--- ---- Local Loopback: Disabled
    @ ---- ---- ---- ---- ---- ---- -00- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- ---1 ---- Mode fault detection: Disabled
    @ ---- ---- ---- ---- ---- ---- ---- 1--- DIV32: SPI clocks at MCK/32
    @ ---- ---- ---- ---- ---- ---- ---- -0-- PCSDEC: Chip selects directly connected
    @ ---- ---- ---- ---- ---- ---- ---- --0- PS: Fixed Peripheral select
    @ ---- ---- ---- ---- ---- ---- ---- ---1 MSTR: SPI is in Master mode
```

```
    @ 0000 0000 0000 1110 0000 0000 0001 1001
    .equ    SPI_MR_VAL, 0x000E0019

    @ 0000 0000 ---- ---- ---- ---- ---- ---- DLYBCT: 0
    @ ---- ---- 0000 0000 ---- ---- ---- ---- DLYBS: 0
    @ ---- ---- ---- ---- 0000 0100 ---- ---- SCBR: SPCK Baudrate = MCK/(2*4) = 5.5 MHz. Device
    max is 20 MHz
    @ ---- ---- ---- ---- ---- ---- 0000 ---- BITS: 8 bits
    @ ---- ---- ---- ---- ---- ---- ---- 00-- Reserved
    @ ---- ---- ---- ---- ---- ---- ---- --0- NCPHA: 1
    @ ---- ---- ---- ---- ---- ---- ---- ---1 CPOL: 1
    @ 0000 0000 0000 0000 1111 0100 0000 0001
    .equ    SPI_CSR0_VAL, 0x00000401

    @ 0000 0000 0000 0000 0000 0000 ---- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- 1000 0000 Only enable ENDTX
    @ 0000 0000 0000 0000 0000 0000 0010 0000
    .equ    SPI_IER_VAL, 0x00000080

    @ 0000 0000 0000 0000 0000 00-- ---- ---- Reserved
    @ ---- ---- ---- ---- ---- --01 ---- ---- Enable Transmit
    @ ---- ---- ---- ---- ---- ---- 0000 00-- Reserved
    @ ---- ---- ---- ---- ---- ---- ---- --10 Disable Receive (no receiving for display)
    @ 0000 0000 0000 0000 0000 0001 0000 0010

    .equ    SPI_DMA_ENABLE, 0x00000101

    @ 0000 0000 0000 0000 0000 00-- ---- ---- Reserved
    @ ---- ---- ---- ---- ---- --10 ---- ---- Disable Transmit
    @ ---- ---- ---- ---- ---- ---- 0000 00-- Reserved
    @ ---- ---- ---- ---- ---- ---- ---- --10 Disable Receive
    @ 0000 0000 0000 0000 0000 0001 0000 0010

    .equ    SPI_DMA_DISABLE, 0x00000202


    @ 0000 0000 0000 0000 0000 0000 0--- ---- Reserved
    @ ---- ---- ---- ---- ---- ---- -11- ---- SCRTYPE: Positive edge
    @ ---- ---- ---- ---- ---- ---- ---0 0--- Reserved
    @ ---- ---- ---- ---- ---- ---- ---- -010 Priority: 2
    @ 0000 0000 0000 0000 0000 0000 0110 0010
    .equ    AIC_SMR13_VAL, 0x00000062

    @ ---- ---- ---- ---- ---- ---- ---- ----


@State definitions for the display redrawing

    .equ    STATE_COMMANDS, 0
    .equ    STATE_DATA, 1


@NHD-C12832 Definitions

    .equ    NUM_PAGES,      4
    .equ    NUM_COLS,       128
    .equ    NHD_ON,         0xAF        @Enable display command
    .equ    NHD_OFF,        0xAE        @Disable display command
    .equ    NHD_RMW,        0xE0        @Read-modify-write command
    .equ    NHD_RESET,      0xE2        @Reset command
    .equ    NHD_PAGE_PREF,  0xB0        @Prefix for page address set command
    .equ    NHD_COLU_PREF,  0x10        @Prefix for upper column address set command
    .equ    NHD_COLL_PREF,  0x00        @Prefix for lower column address set command


@ASCII Definitions
```

```
    .equ    ASCII_DOT,      0x2E        @ '.' character
    .equ    ASCII_NULL,     0x00        @ ASCII null termination

@Miscellaneous definitions
    .equ    NUM_OCTETS_IP,  4
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@ Display.s
@
@ Description: Contains code for controlling the display component of
@ the EE52 VoIP Project
@
@ Table of Contents:
@   - display_init: Call to initialize the shared variables and peripherals
@                   for interfacing with the C12832 Display
@   - display_memory_addr: Displays the passed memory address on display
@   - display_IP: Displays the passed IP address on display
@   - display_status: Displays the passed status on display
@   - redraw: Redraws the display buffer
@   - queueDisplayCommand: Queues a command to send to the display
@   - setBacklight: Sets the backlight on or off
@   - displayHandler: Handles setting up display DMA stream and also
@                     sending commands to display
@
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017

.include    "macro.inc"
.include    "display.inc"
.include    "pio.inc"
.include    "at91rm9200.inc"
.include    "system.inc"
.include    "interfac.inc"


.arm
.text

@ display_init
@
@ Description: Initializes the display
@
@ Operational Description: The PIO pins are initialized,
@                          and then a series of commands
@                          are queued up to initialize the
@                          display. Then, the display handler
@                          is installed and kickstarted by starting
@                          the display output DMA engine.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
```

```
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global display_init
display_init:
    mSTARTFNC

    @First, initialize the PIO pins
    mSET_HREG   PMC_PCER,   (1 << 13)
    LDR     r0, =(0xE)          @MOSI, SPCK, NPCS0
    LDR     r1, =PIO_A
    LDR     r2, =(PIO_PERA | PIO_OUTPUT)
    BL      configPIOPin

    LDR     r0, =(1 << DISP_A0) | (1 << DISP_BCKLIGHT) | (1 << DISP_RST)
    LDR     r1, =PIO_A
    LDR     r2, =(PIO_NORM | PIO_OUTPUT)
    BL      configPIOPin

    mSET_HREG PIOA_CODR,    (1 << DISP_RST) @Reset display

    @Need to wait for the display to reset, this loop
    @introduces a delay to allow the display to reset
    LDR r0, =0xFFFF
count:
    SUB r0, #1
    CMP r0, #0
    BNE count
    @Done with delay

    mSET_HREG PIOA_SODR,    (1 << DISP_RST) @Stop reset signal to display

    @Set backlight on
    LDR     r0, =TRUE
    BL      setBacklight

    @Enqueue all of the commands for initializing display now
    LDR     r0, =NHD_RESET
    BL      queueDisplayCommand
    @LDR        r0, =NHD_ON
    @BL     queueDisplayCommand
    @LDR        r0, =0x2F
    @BL     queueDisplayCommand
    @LDR        r0, =0x26
    @BL     queueDisplayCommand
    @LDR        r0, =NHD_RMW
    @BL     queueDisplayCommand
    @LDR        r0, =0xA4
    @BL     queueDisplayCommand
    @LDR        r0, =0x81
    @BL     queueDisplayCommand
    @LDR        r0, =0x2F
    @BL     queueDisplayCommand
```

```
    LDR     r0, =0xAE
    BL      queueDisplayCommand
    LDR     r0, =0xA2
    BL      queueDisplayCommand

    LDR     r0, =0xA0
    BL      queueDisplayCommand
    LDR     r0, =0xC8
    BL      queueDisplayCommand

    LDR     r0, =0x22
    BL      queueDisplayCommand
    LDR     r0, =0x2F
    BL      queueDisplayCommand

    LDR     r0, =0x40
    BL      queueDisplayCommand
    LDR     r0, =0xAF
    BL      queueDisplayCommand

    LDR     r0, =0x81
    BL      queueDisplayCommand
    LDR     r0, =0x17
    BL      queueDisplayCommand

    LDR     r0, =0xA6
    BL      queueDisplayCommand

    @Setup SPI
    mSET_HREG   SPI_CR, SPI_CR_RESET
    mSET_HREG   SPI_MR, SPI_MR_VAL
    mSET_HREG   SPI_CSR0, SPI_CSR0_VAL

    @Install interrupt for when DMA finished to display
    mSET_HREG   AIC_SVR13, displayHandler
    mSET_HREG   AIC_SMR13, AIC_SMR13_VAL
    mSET_HREG   AIC_IECR, (1 << 13)

    @Finish setting up SPI and enable DMA interrupt
    mSET_HREG   SPI_CR, SPI_CR_EN
    mSET_HREG   SPI_PTCR,   SPI_DMA_ENABLE  @Enable the DMA controller
    mSET_HREG   SPI_IER, SPI_IER_VAL
    mRETURNFNC

@ display_memory_addr
@
@ Description: Displays the passed memory address on the display.
@
@ Operational Description: registers 4-7 are pushed to stack so they
@                          can be used. The string is composed by looking
@                          up each character in a hexToAscii table.
@                          After the string is finished being written,
@                          the buffers are redrawn using the redraw
@                          function, and then the method returns.
@                          The display is redrawn the next time the DMA
@                          engine writes the buffer to the display
@
@
@ Arguments: r0 - unsigned int of address to display
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: TextMessageR2[W] - The message displayed on 2nd line
@                                      is replaced with the memory address
```

```
@                                       string representation generated
@                                       by this function.
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name              Comment              Date
@ Will Werst         Initial version     Some lonely night around 6/10/17

.global display_memory_addr
display_memory_addr:
    mSTARTFNC

    PUSH    {r4-r7}
    LDR r5, =TextMessageR2          @String to write to
    LDR r6, =0                      @ASCII string position


    MOV r7, r0                      @Store memory address to use later
    @Get tens digit
    LDR r1, =10
    MOV r0, r7
    BL  divide
    LDR r1, =hexToASCII
    LDRB    r0, [r1, r0]
    STRB    r0, [r5, r6]
    ADD     r6, #1

    @Remove tens digit value from memory address
    LDR r1, =10
    MOV r0, r7
    BL  mod
    MOV r7, r0                      @r7 now contains original address with tens digit 0

    @Get ones digit
    LDR r1, =hexToASCII
    LDRB    r0, [r1, r0]
    STRB    r0, [r5, r6]
    ADD     r6, #1

    @Add null termination to string
    LDRB    r0, =ASCII_NULL
    STRB    r0, [r5, r6]


    BL redraw                       @Redraw the buffer

    POP {r4-r7}
```

```
    mRETURNFNC

@ display_IP
@
@ Description:
@
@ Operational Description: registers 4-7 are pushed to stack so they
@                          can be used. The octets are then extracted from
@                          the input IP address to be displayed, and each
@                          octet is pushed to the stack. Then, each octet
@                          is popped off the stack in a loop and added to
@                          the 2nd row on display string.
@                          Finally, the redraw method is used to update
@                          the display buffers.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: TextMessageR2[W] - The message displayed on 2nd line
@                                      is replaced with the IP address
@                                      string representation generated
@                                      by this function.
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global display_IP
display_IP:
    mSTARTFNC

    PUSH {r4-r7}
    @Get 4th octet
    AND r1, r0, #0xFF
    LSR r0, #8
    PUSH {r1}

    @Get 3rd octet
    AND r1, r0, #0xFF
    LSR r0, #8
    PUSH {r1}

    @Get 2nd octet
```

```
    AND r1, r0, #0xFF
    LSR r0, #8
    PUSH {r1}

    @Get 1st octet
    AND r1, r0, #0xFF
    LSR r0, #8
    PUSH {r1}

    LDR r4, =NUM_OCTETS_IP
    LDR r5, =TextMessageR2          @String to write to
    LDR r6, =0                      @ASCII string position
OctetToASCII:
    POP {r0}                        @Get next octet
    MOV r7, r0                      @Store octet so can operate on it and recover later


    @Get hundreds digit
    LDR r1, =100
    MOV r0, r7
    BL  divide
    LDR r1, =hexToASCII
    LDRB    r0, [r1, r0]
    STRB    r0, [r5, r6]
    ADD     r6, #1

    @Remove hundreds digit value from octet
    LDR r1, =100
    MOV r0, r7
    BL  mod
    MOV r7, r0                      @r7 now contains original octet with hundreds digit 0

    @Get tens digit
    LDR r1, =10
    MOV r0, r7
    BL  divide
    LDR r1, =hexToASCII
    LDRB    r0, [r1, r0]
    STRB    r0, [r5, r6]
    ADD     r6, #1

    @Remove tens digit value from octet
    LDR r1, =10
    MOV r0, r7
    BL  mod
    MOV r7, r0                      @r7 now contains original octet with hundreds and tens
    digit 0

    @Get ones digit
    LDR r1, =hexToASCII
    LDRB    r0, [r1, r0]
    STRB    r0, [r5, r6]
    ADD     r6, #1

    @Add '.' character to terminate string
    LDRB    r0, =ASCII_DOT
    STRB    r0, [r5, r6]
    ADD     r6, #1

    SUB r4, #1
    CMP r4, #0
    BHI OctetToASCII

    @Add null termination to string
    LDRB    r0, =ASCII_NULL
    SUB     r6, #1                  @Left a '.' character at end, will overwrite with null.
```

```
    STRB    r0, [r5, r6]


    BL redraw

    POP {r4-r7}
    mRETURNFNC


@ display_status
@
@ Description: Displays the passed display status
@
@ Operational Description: The string corresponding to the passed display status
@                          is looked up and rendered to the display buffer. If
@                          the status is STATUS_IDLE, the rest of the display
@                          is cleared.
@
@ Arguments: r0 - status to display
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: TextMessageR1[W] - The message displayed on 1nd line
@                                      is replaced with the status
@                                      string representation generated
@                                      by this function.
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global display_status
display_status:
    mSTARTFNC
    CMP     r0, #STATUS_IDLE            @Check if status is idle
    LDREQ     r1, =TextMessageR2        @And if so, clear the display
    LDREQ     r2, =ASCII_NULL
    @Clear the 2nd row of display
    STREQB    r2, [r1]

    @Load the status message to TextMessageR1
    LDR     r1, =StatusMessages
    LDR     r1, [r1, r0, LSL #2]
    mSTOREFROMREG r1, r0, TextMessageR1
```

```
    @Redraw the display
    BL  redraw
    mRETURNFNC


@ redraw
@
@ Description: The display buffer is redrawn. Call
@              after changing TextMessageR1 or TextMessageR2.
@
@ Operational Description: The display is cleared using the clear_displaybuffer
@                          C method in displayrenderer.c. Then, the 1st and 2nd
@                          row are rendered using the render_displaybuffer
@                          C method in displayrenderer.c
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global redraw
redraw:
    mSTARTFNC
    LDR r0, =DispBuffer
    LDR r1, =(NUM_COLS*NUM_PAGES)
    BL  clear_displaybuffer

    LDR r0, =TextMessageR1
    LDR r0, [r0]
    LDR r1, =DispBuffer
    LDR r2, =NUM_COLS
    BL render_displaybuffer

    LDR r0, =TextMessageR2
    LDR r1, =(DispBuffer+NUM_COLS)
    LDR r2, =NUM_COLS
    BL render_displaybuffer
```

```
    mRETURNFNC


@ queueDisplayCommand
@
@ Description: queues the command passed in r0 to be sent to the display.
@
@ Operational Description: Interrupts are disabled since this accesses
@                          the command queue which is also accessed in
@                          the displayHandler. The command is enqueued if
@                          there is space. If the command was enqueued
@                          successfully, the method exits with TRUE, else FALSE.
@
@
@ Arguments: r0 - command to send over SPI
@
@ Return values: r0 - TRUE if added successfully, FALSE if not added.
@
@ Local variables: None
@
@ Shared variables: CommandQueueSize[RW] - read to check queue size,
@                                          and incremented after adding queue element.
@                   ActiveCommandQueue[W] - command is enqueued.
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: If queue is full, the method exits with FALSE
@
@ Algorithms: None
@
@ Data Structures: Queue
@
@ Limitations: None known
@
@ Registers Changed (other than r0-r3): None
@
@ Known Bugs: None known
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     6/22/2017

queueDisplayCommand:
    mSTARTFNC
    mSTARTCRITCODE                      @Enter critical code, r7 used by this macro
    mLOADTOREG  r1, CommandQueueSize
    CMP r1, #COM_QUEUE_LENGTH
    BLO addCommandToQ
    @B  CommandQFull
CommandQFull:
    LDR r0, =FALSE
    B   sDCEndCritCode
addCommandToQ:
    LDR r2, =ActiveCommandQueue         @Load pointer to activeCommandQueue
    LDR r2, [r2]                        @Load the queue pointed to by activeCommandQueue
    STRB r0, [r2,r1]
    ADD r1, #1
    mSTOREFROMREG r1, r0, CommandQueueSize
    LDR r0, =TRUE
sDCEndCritCode:
```

```
    mENDCRITCODE                        @Exit critical code, r7 used by this macro

    mRETURNFNC

@ setBacklight
@
@ Description: Enables or disables the backlight.
@
@ Operational Description: The backlight is set on or off.
@
@ Arguments: r0 - TRUE if backlight should be enabled, otherwise backlight disabled.
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: Backlight of display
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None known
@
@ Registers Changed (besides r0-r3): None
@
@ Known Bugs: None known
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     6/22/2017

setBacklight:
    mSTARTFNC
    CMP     r0, #TRUE           @Check which action to take
    BEQ     backlightOn         @if true, enable backlight
    BNE     backlightOff        @else, disable backlight
backlightOn:
    mSET_HREG   PIOA_SODR,  (1 << DISP_BCKLIGHT)
    B       endSetBacklight
backlightOff:
    mSET_HREG   PIOA_CODR,  (1 << DISP_BCKLIGHT)
    B       endSetBacklight
endSetBacklight:
    mRETURNFNC


@ displayHandler
@
@ Description: Handles sending commands to the display
@
@ Operational Description: This interrupt routine handles a state machine that
@                           updates the display. This state machine has two
@                           states: Commands, and Data. The different states
@                           are handled as follows:
@                           Commands: The ActiveCommandQueue is switched to the other queue.
```

```
The command to
@                                       switch to the next page of data is enqueued if possible.
@                                       If there is space for this command, then the state
transitions
@                                       to Data next, and the current page is updated to the
next page, else the state stays at Commands.
@                                       The display is transitioned to command mode (DISP_A0
pin is set to 0 in PIO). Then, the DMA controller
@                                       is set to transmit all commands in the now-inactive
command queue.
@                           Data: The DMA controller is setup to transmit the next page of data
from DispBuffer.
@
@
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: ActiveCommandQueue[RW] - Read from to send commands, and toggled back and
forth between
@                                           queue 1 and queue 2.
@
@                   displayHandlerState[RW] - Checked to determine whether sending data or
commands.
@                   displayCurPage[RW] - Checked to determine which page in display buffer to
send next.
@                   displayBuffer[R] - Data is read from here to send to display
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: SPI to display
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: Queues
@
@ Limitations: None
@
@ Registers Changed: None
@
@ Known Bugs: None known
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     6/22/2017

displayHandler:
    mSTARTINT
    mSET_HREG   SPI_PTCR,   SPI_DMA_DISABLE @Disable the DMA controller
    mSET_HREG   SPI_CR, SPI_CR_DIS
waitSPIDIS:
    mLOADTOREG  r0, SPI_SR
    TST r0, #0x10000
    BNE waitSPIDIS
    mLOADTOREG  r0, displayHandlerState
    CMP r0, #STATE_COMMANDS
    BEQ stateCommand
```

```
    CMP r0, #STATE_DATA
    BEQ stateData
    B   endDisplayHandler                   @Should never hit this state
stateCommand:
    LDR r0, =(NHD_COLL_PREF | 0x0)
    BL  queueDisplayCommand
    LDR r0, =(NHD_COLU_PREF | 0x0)
    BL  queueDisplayCommand
    mLOADTOREG  r0, displayCurPage          @Add the page address set command
    ORR r0, r0, #NHD_PAGE_PREF
    BL  queueDisplayCommand
    CMP r0, #TRUE                           @Check if command added
    LDREQ   r0, =STATE_DATA                 @If it was, transition states
    LDREQ   r1, =displayHandlerState        @Else, the state is left as STATE_COMMANDS
    STREQ   r0, [r1]
    LDR r0, =ActiveCommandQueue             @Load the address of activeCommandQueue pointer
    LDR r1, [r0]                            @Dereference activeCommandQueue to get
                                            @pointer to active queue's start.
    PUSH {r1}                               @Save pointer to current active queue for later.
    LDR r2, =CommandQueue1                  @Load pointer to CommandQeueu1 to
                                            @compare to active command queue
    CMP r1, r2                              @Compare the queues
    LDREQ   r2, =CommandQueue2              @If active queue is 1, switch to queue 2.
                                            @Else, queue 2 is active and want to switch to
                                            queue 1
    STR     r2, [r0]                        @switch to the new queue

    @Setup DMA for sending commands
    mSET_HREG   PIOA_CODR,  (1 << DISP_A0)  @Clear A0 (sending commands)
    POP {r1}                                @Get the queue of commands to send
    mSTOREFROMREG   r1, r0, SPI_TPR     @Set pointer to now inactive command queue
    mLOADTOREG  r1, CommandQueueSize        @Set count of bytes to send to command queue size
    mSTOREFROMREG   r1, r0, SPI_TCR

    LDR r1, =CommandQueueSize               @clear command queue size since now using different
    queue
    LDR r0, =0
    STR r0, [r1]
    B   endDisplayHandler
stateData:

    @Setup DMA for sending data
    mSET_HREG   PIOA_SODR,  (1 << DISP_A0)  @Set A0 (sending data)
    mLOADTOREG r0, displayCurPage           @Calculate the pointer to the current
                                            @buffer page

    LDR r1, =NUM_COLS
    MUL r0, r0, r1
    LDR r1, =DispBuffer
    ADD r0, r0, r1
    mSTOREFROMREG   r0, r2, SPI_TPR         @Set pointer to display buffer
    LDR r1, =NUM_COLS
    mSTOREFROMREG   r1, r2, SPI_TCR         @Set length of DMA

    LDR r0, =STATE_COMMANDS                 @Transition states
    LDR r1, =displayHandlerState
    STR r0, [r1]
    mLOADTOREG r0, displayCurPage           @Calculate the pointer to the current
                                            @buffer page
    ADD r0, #1
    CMP r0, #NUM_PAGES
    LDREQ   r0, =0
    mSTOREFROMREG   r0, r1, displayCurPage
    @B endDisplayHandler
endDisplayHandler:
    mSET_HREG   SPI_CR, SPI_CR_EN
```

```
    mSET_HREG   SPI_PTCR,   SPI_DMA_ENABLE  @Enable the DMA controller
    mRETURNINT


.data
.balign 4

@Pointer to null-terminated ASCII string
displayHandlerState:
    .word STATE_COMMANDS
displayUpdated:
    .word TRUE
displayCurPage:
    .word 0

hexToASCII:                 @Table to map hex bytes to ASCII characters
    .byte   0x30
    .byte   0x31
    .byte   0x32
    .byte   0x33
    .byte   0x34
    .byte   0x35
    .byte   0x36
    .byte   0x37
    .byte   0x38
    .byte   0x39
    .byte   0x41
    .byte   0x42
    .byte   0x43
    .byte   0x44
    .byte   0x45
    .byte   0x46

.balign 4
TextMessageR1:
    .word   St_IDLE_MES
TextMessageR2:
    .skip   22              @Max 21 characters on display plus null termination
.balign 4
StatusMessages:
    .word   St_IDLE_MES
    .word   St_OFFHOOK_MES
    .word   St_RINGING_MES
    .word   St_CONNECTING_MES
    .word   St_CONNECTED_MES
    .word   St_SET_IP_MES
    .word   St_SET_SUBNET_MES
    .word   St_SET_GATEWAY_MES
    .word   St_MEM_SAVE_MES
    .word   St_MEM_RECALL_MES
    .word   St_RECALLED_MES
    .word   St_ILLEGAL_MES

St_IDLE_MES:
    .asciz  "Idle"
St_OFFHOOK_MES:
    .asciz  "Off Hook"
St_RINGING_MES:
    .asciz  "Ringing"
St_CONNECTING_MES:
    .asciz  "Connecting"
St_CONNECTED_MES:
    .asciz  "Connected"
St_SET_IP_MES:
    .asciz  "Set IP"
St_SET_SUBNET_MES:
```

```
    .asciz  "Set Subnet"
St_SET_GATEWAY_MES:
    .asciz  "Set Gateway"
St_MEM_SAVE_MES:
    .asciz  "Memory Save"
St_MEM_RECALL_MES:
    .asciz  "Memory Recall"
St_RECALLED_MES:
    .asciz  "Recalled Message"
St_ILLEGAL_MES:
    .asciz  "Illegal Message"


CommandQueueSize:
    .word 0x00000000
ActiveCommandQueue:
    .word CommandQueue1



CommandQueue1:
    .skip   (COM_QUEUE_LENGTH)
CommandQueue2:
    .skip   (COM_QUEUE_LENGTH)
DispBuffer:
    .skip (NUM_COLS*NUM_PAGES)
.balign 4
.end
```

```
//Done
/*********************************************************************/
/*                                                                   */
/*                          Display Renderer                         */
/*                       5x7 Dot Matrix Codes                        */
/*              taken from Digital Oscilloscope Project              */
/*                             EE/CS 52                              */
/*                                                                   */
/*********************************************************************/

/*
   This file contains the render_displaybuffer function. It takes as input a pointer to
   the start of the display buffer to write to, and an ASCII string to render there

   Revision History
      5/27/08  Glen George       Initial revision of ascii_patterns(from
                                 3/10/95 version of char57.asm).
      6/10/17  Will Werst        Initial code
*/

const unsigned char ascii_char_patterns[] = {

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x00)            */
    0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x04,   /* up arrow (0x01)          */
    0x04, 0x04, 0x04, 0x04, 0x15, 0x0E, 0x04,   /* down arrow (0x02)        */
    0x00, 0x04, 0x08, 0x1F, 0x08, 0x04, 0x00,   /* left arrow (0x03)        */
    0x00, 0x11, 0x11, 0x11, 0x1B, 0x14, 0x10,   /* greek u (mu) (0x04)      */
    0x00, 0x04, 0x02, 0x1F, 0x02, 0x04, 0x00,   /* right arrow (0x05)       */
    0x00, 0x11, 0x0A, 0x04, 0x0A, 0x11, 0x00,   /* multiply symbol (0x06)   */
    0x00, 0x04, 0x00, 0x1F, 0x00, 0x04, 0x00,   /* divide symbol (0x07)     */
    0x04, 0x04, 0x1F, 0x04, 0x04, 0x00, 0x1F,   /* plus/minus symbol (0x08) */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x09)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x0A)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x0B)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x0C)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x0D)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x0E)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x0F)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x10)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x11)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x12)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x13)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x14)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x15)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x16)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x17)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x18)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x19)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x1A)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x1B)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x1C)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x1D)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x1E)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* UNUSED (0x1F)            */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* space (0x20)             */
    0x04, 0x04, 0x04, 0x04, 0x04, 0x00, 0x04,   /* !                        */
    0x0A, 0x0A, 0x0A, 0x00, 0x00, 0x00, 0x00,   /* "                        */
    0x0A, 0x0A, 0x1F, 0x0A, 0x1F, 0x0A, 0x0A,   /* #                        */
    0x04, 0x0F, 0x14, 0x0E, 0x05, 0x1E, 0x04,   /* $                        */
    0x18, 0x19, 0x02, 0x04, 0x08, 0x13, 0x03,   /* %                        */
    0x08, 0x14, 0x14, 0x08, 0x15, 0x12, 0x0D,   /* &                        */
    0x0C, 0x0C, 0x08, 0x10, 0x00, 0x00, 0x00,   /* '                        */
    0x02, 0x04, 0x08, 0x08, 0x08, 0x04, 0x02,   /* (                        */
    0x08, 0x04, 0x02, 0x02, 0x02, 0x04, 0x08,   /* )                        */
    0x04, 0x15, 0x0E, 0x1F, 0x0E, 0x15, 0x04,   /* *                        */
    0x00, 0x04, 0x04, 0x1F, 0x04, 0x04, 0x00,   /* +                        */
```

```
    0x00, 0x00, 0x00, 0x0C, 0x0C, 0x08, 0x10,   /* ,                */
    0x00, 0x00, 0x00, 0x1F, 0x00, 0x00, 0x00,   /* -                */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C,   /* .                */
    0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x00,   /* /                */
    0x0E, 0x11, 0x13, 0x15, 0x19, 0x11, 0x0E,   /* 0                */
    0x04, 0x0C, 0x04, 0x04, 0x04, 0x04, 0x0E,   /* 1                */
    0x0E, 0x11, 0x01, 0x0E, 0x10, 0x10, 0x1F,   /* 2                */
    0x0E, 0x11, 0x01, 0x06, 0x01, 0x11, 0x0E,   /* 3                */
    0x02, 0x06, 0x0A, 0x12, 0x1F, 0x02, 0x02,   /* 4                */
    0x1F, 0x10, 0x1E, 0x01, 0x01, 0x11, 0x0E,   /* 5                */
    0x06, 0x08, 0x10, 0x1E, 0x11, 0x11, 0x0E,   /* 6                */
    0x1F, 0x01, 0x02, 0x04, 0x08, 0x10, 0x10,   /* 7                */
    0x0E, 0x11, 0x11, 0x0E, 0x11, 0x11, 0x0E,   /* 8                */
    0x0E, 0x11, 0x11, 0x0F, 0x01, 0x02, 0x0C,   /* 9                */
    0x00, 0x0C, 0x0C, 0x00, 0x0C, 0x0C, 0x00,   /* :                */
    0x0C, 0x0C, 0x00, 0x0C, 0x0C, 0x08, 0x10,   /* ;                */
    0x02, 0x04, 0x08, 0x10, 0x08, 0x04, 0x02,   /* <                */
    0x00, 0x00, 0x1F, 0x00, 0x1F, 0x00, 0x00,   /* =                */
    0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08,   /* >                */
    0x0E, 0x11, 0x01, 0x02, 0x04, 0x00, 0x04,   /* ?                */
    0x0E, 0x11, 0x01, 0x0D, 0x15, 0x15, 0x0E,   /* @                */
    0x04, 0x0A, 0x11, 0x11, 0x1F, 0x11, 0x11,   /* A                */
    0x1E, 0x09, 0x09, 0x0E, 0x09, 0x09, 0x1E,   /* B                */
    0x0E, 0x11, 0x10, 0x10, 0x10, 0x11, 0x0E,   /* C                */
    0x1E, 0x09, 0x09, 0x09, 0x09, 0x09, 0x1E,   /* D                */
    0x1F, 0x10, 0x10, 0x1C, 0x10, 0x10, 0x1F,   /* E                */
    0x1F, 0x10, 0x10, 0x1C, 0x10, 0x10, 0x10,   /* F                */
    0x0F, 0x10, 0x10, 0x13, 0x11, 0x11, 0x0F,   /* G                */
    0x11, 0x11, 0x11, 0x1F, 0x11, 0x11, 0x11,   /* H                */
    0x0E, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0E,   /* I                */
    0x01, 0x01, 0x01, 0x01, 0x01, 0x11, 0x0E,   /* J                */
    0x11, 0x12, 0x14, 0x18, 0x14, 0x12, 0x11,   /* K                */
    0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x1F,   /* L                */
    0x11, 0x1B, 0x15, 0x15, 0x11, 0x11, 0x11,   /* M                */
    0x11, 0x19, 0x15, 0x13, 0x11, 0x11, 0x11,   /* N                */
    0x0E, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0E,   /* O                */
    0x1E, 0x11, 0x11, 0x1E, 0x10, 0x10, 0x10,   /* P                */
    0x0E, 0x11, 0x11, 0x11, 0x15, 0x12, 0x0D,   /* Q                */
    0x1E, 0x11, 0x11, 0x1E, 0x14, 0x12, 0x11,   /* R                */
    0x0E, 0x11, 0x10, 0x0E, 0x01, 0x11, 0x0E,   /* S                */
    0x1F, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,   /* T                */
    0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0E,   /* U                */
    0x11, 0x11, 0x11, 0x0A, 0x0A, 0x04, 0x04,   /* V                */
    0x11, 0x11, 0x11, 0x11, 0x15, 0x1B, 0x11,   /* W                */
    0x11, 0x11, 0x0A, 0x04, 0x0A, 0x11, 0x11,   /* X                */
    0x11, 0x11, 0x0A, 0x04, 0x04, 0x04, 0x04,   /* Y                */
    0x1F, 0x01, 0x02, 0x04, 0x08, 0x10, 0x1F,   /* Z                */
    0x0E, 0x08, 0x08, 0x08, 0x08, 0x08, 0x0E,   /* [                */
    0x00, 0x10, 0x08, 0x04, 0x02, 0x01, 0x00,   /* \                */
    0x0E, 0x02, 0x02, 0x02, 0x02, 0x02, 0x0E,   /* ]                */
    0x04, 0x0A, 0x11, 0x00, 0x00, 0x00, 0x00,   /* ^                */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F,   /* _                */
    0x06, 0x06, 0x04, 0x02, 0x00, 0x00, 0x00,   /* `                */
    0x00, 0x00, 0x0E, 0x01, 0x0F, 0x11, 0x0F,   /* a                */
    0x10, 0x10, 0x16, 0x19, 0x11, 0x19, 0x16,   /* b                */
    0x00, 0x00, 0x0E, 0x11, 0x10, 0x11, 0x0E,   /* c                */
    0x01, 0x01, 0x0D, 0x13, 0x11, 0x13, 0x0D,   /* d                */
    0x00, 0x00, 0x0E, 0x11, 0x1F, 0x10, 0x0E,   /* e                */
    0x02, 0x05, 0x04, 0x0E, 0x04, 0x04, 0x04,   /* f                */
    0x0D, 0x13, 0x13, 0x0D, 0x01, 0x11, 0x0E,   /* g                */
    0x10, 0x10, 0x16, 0x19, 0x11, 0x11, 0x11,   /* h                */
    0x04, 0x00, 0x0C, 0x04, 0x04, 0x04, 0x0E,   /* i                */
    0x01, 0x00, 0x01, 0x01, 0x01, 0x11, 0x0E,   /* j                */
    0x10, 0x10, 0x12, 0x14, 0x18, 0x14, 0x12,   /* k                */
    0x0C, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0E,   /* l                */
    0x00, 0x00, 0x1A, 0x15, 0x15, 0x15, 0x15,   /* m                */
```

```
    0x00, 0x00, 0x16, 0x19, 0x11, 0x11, 0x11,   /* n                        */
    0x00, 0x00, 0x0E, 0x11, 0x11, 0x11, 0x0E,   /* o                        */
    0x16, 0x19, 0x11, 0x19, 0x16, 0x10, 0x10,   /* p                        */
    0x0D, 0x13, 0x11, 0x13, 0x0D, 0x01, 0x01,   /* q                        */
    0x00, 0x00, 0x16, 0x19, 0x10, 0x10, 0x10,   /* r                        */
    0x00, 0x00, 0x0F, 0x10, 0x0E, 0x01, 0x1E,   /* s                        */
    0x04, 0x04, 0x1F, 0x04, 0x04, 0x05, 0x02,   /* t                        */
    0x00, 0x00, 0x11, 0x11, 0x11, 0x13, 0x0D,   /* u                        */
    0x00, 0x00, 0x11, 0x11, 0x11, 0x0A, 0x04,   /* v                        */
    0x00, 0x00, 0x11, 0x11, 0x15, 0x15, 0x0A,   /* w                        */
    0x00, 0x00, 0x11, 0x0A, 0x04, 0x0A, 0x11,   /* x                        */
    0x11, 0x11, 0x11, 0x0F, 0x01, 0x11, 0x0E,   /* y                        */
    0x00, 0x00, 0x1F, 0x02, 0x04, 0x08, 0x1F,   /* z                        */
    0x02, 0x04, 0x04, 0x08, 0x04, 0x04, 0x02,   /* {                        */
    0x04, 0x04, 0x04, 0x00, 0x04, 0x04, 0x04,   /* |                        */
    0x08, 0x04, 0x04, 0x02, 0x04, 0x04, 0x08,   /* }                        */
    0x08, 0x15, 0x02, 0x00, 0x00, 0x00, 0x00,   /* ~                        */
    0x0A, 0x15, 0x0A, 0x15, 0x0A, 0x15, 0x0A    /* DEL (0x7F)               */

};



/* library include files */


/* local include files */
    /* none */

/*
   render_displaybuffer

   Description: Renders the passed string to the passed buffer.

   Operation: The buffer is a sequence of bytes that specify
              how each column of pixels should be set, with each bit
              in a byte setting one pixel in a column. The code goes
              through all the columns, and picks out the appropriate
              bits from the ascii_char_patterns.

   Arguments: *string - pointer to start of null-terminated string
              *buffer - pointer to start of display buffer
              length - length of string
   Return Value: None

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Shared Variables:

   Author:           Will Werst
   Last Modified:    June 23, 2017

*/

void render_displaybuffer(char *string, char *buffer, int length){
    int col;
    for (col = 0; col < length; col++){
        /*if (col % 6 == 0){
            buffer[col] = 0x00;
            continue;
```

```
        }*/
        int cur_str_pos = (col / 6);
        char cur_char = string[cur_str_pos];
        if (cur_char == 0x00){
            break;
        }
        int row;
        for (row = 0; row < 7; row++){
            if ((ascii_char_patterns[cur_char*7 + row] & (0x10 >> (col % 6))) != 0){         //
            If the current value is blank
                buffer[col] = buffer[col] ^ (0x1 << row);
            }
        }
    }
}

// Clears the passed buffer
void clear_displaybuffer(char *buffer, int length){
    int col;
    for (col = 0; col < 512; col++){
        buffer[col] = 0x00;
    }
}

// Does what it says it does
int divide(int num, int den){
    return num / den;
}

// Does what it says it does
int mod(int num, int den){
    return num % den;
}
```

```
@Done
@/*****************************************************************************/
@/*                                                                           */
@/*                                INTERFAC.H                                 */
@/*                           Interface Definitions                           */
@/*                               Include File                                */
@/*                           VoIP Telephone Project                          */
@/*                                 EE/CS 52                                  */
@/*                                                                           */
@/*****************************************************************************/

@   This file contains the constants for interfacing between the C code and
@   the assembly code/hardware.


@   Revision History:
@   Name        Comment                          Date
@   Glen George Initial revision.                6/3/06
@   Glen George Added ETHER_INTF definition.     6/7/06
@   Glen George Added MAC_ADDR_H and             3/8/11
@                   MAC_ADDR_L definitions.
@   Will Werst  Changed values to match          Some lonely night around 6/10/17
@                   my specification


        .equ  MEMORY_SIZE,     32
        .equ  SAMPLE_RATE,    8000

        .equ  MAC_ADDR_H,      0xEA09
        .equ  MAC_ADDR_L,      0x87654321

        @.equ  ETHER_INTF,      "et0"

        .equ  KEY_0,           0
        .equ  KEY_1,           1
        .equ  KEY_2,           2
        .equ  KEY_3,           3
        .equ  KEY_4,           4
        .equ  KEY_5,           5
        .equ  KEY_6,           6
        .equ  KEY_7,           7
        .equ  KEY_8,           8
        .equ  KEY_9,           9
        .equ  KEY_ESC,         10
        .equ  KEY_BACKSPACE,   11
        .equ  KEY_SEND,        12
        .equ  KEY_OFFHOOK,     13
        .equ  KEY_ONHOOK,      14
        .equ  KEY_SET_IP,      15
        .equ  KEY_SET_SUBNET,  16
        .equ  KEY_SET_GATEWAY, 17
        .equ  KEY_MEM_SAVE,    18
        .equ  KEY_MEM_RECALL,  19
        .equ  KEY_ILLEGAL,     20

        .equ  STATUS_IDLE,          0
        .equ  STATUS_OFFHOOK,       1
        .equ  STATUS_RINGING,       2
        .equ  STATUS_CONNECTING,    3
        .equ  STATUS_CONNECTED,     4
        .equ  STATUS_SET_IP,        5
        .equ  STATUS_SET_SUBNET,    6
        .equ  STATUS_SET_GATEWAY,   7
        .equ  STATUS_MEM_SAVE,      8
        .equ  STATUS_MEM_RECALL,    9
        .equ  STATUS_RECALLED,      10
```

```
        .equ   STATUS_ILLEGAL,      11

        .equ   AUDIO_BUFLEN,    256
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                   keypad.inc
@                       Contains definitions used by keypad.s
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ Description: Defines register values used by the keypad
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017



    .equ RDY_PIN,    26                 @Ready pin location for interrupts
    .equ MAXDEBCOUNT,   10000
    .equ  KEY_HANGUPBTN, 21
    .equ  KEY_SHIFT,     22
```

```
@ Shared variables: LastKeypress - initialized to KEY_ILLEGAL
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                    macro.inc
@                   Defines various macros used throughout the code base
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ Description: Defines register values used by the audio system.
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017

@Macro for setting Hardware register at destination with value val
@Registers modified: r0, r1
.macro mSET_HREG dest, value

LDR r0, =\dest
LDR r1, =\value
STR r1, [r0]

.endm

.macro mLOADTOREG reg, mem_loc

    LDR \reg,   =\mem_loc
    LDR \reg,   [\reg]

.endm

.macro mSTOREFROMREG reg, temp_reg, mem_loc

    LDR \temp_reg, =\mem_loc
    STR \reg, [\temp_reg]

.endm

.macro mSTARTFNC
    PUSH {lr}
.endm

.macro mRETURNFNC
    POP {pc}
.endm

.macro mSTARTINT
    PUSH {r0-r3, lr}
.endm

.macro mRETURNINT
    mSET_HREG AIC_EOICR, 1
    POP {r0-r3, lr}
    SUBS pc, lr, #WORD_SIZE
.endm


@ Macros to handle critical code. Note that r7 is used, the value is restored
@ after critical code section, but since the stack is used to store value
@ in critical code start, its value cannot be preserved during the critical
@ code section.
```

```
.macro  mSTARTCRITCODE
    PUSH {r7}
    MRS r7, cpsr
    PUSH {r7}
    ORR r7, r7, #(I_BIT | F_BIT)
    MSR cpsr_c, r7
.endm

.macro mENDCRITCODE
    POP {r7}
    MSR cpsr, r7
    POP {r7}
.endm
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@ Memtest.s
@
@
@
@
@ Description: Contains code for testing the SRAM and DRAM
@              This code is normally ran before copying the system from ROM to SRAM or DRAM
@
@ Table of Contents:
@   - mem_test: Tests the memory address range that is passed.
@               The test erases anything stored here before.
@
@
@
@
@
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017


.include    "at91rm9200.inc"
.include    "system.inc"


.text
.arm

@ mem_test
@
@ Description: Tests memory of passed memory section.
@              In the process of testing memory, this function destroys all data
@              that was in the memory already.
@
@ Operational Description: The code works by writing a sequence of numbers
@                          to successive bytes in the memory that are fairly
@                          random and do not repeat on any 2^n periodicity, and
@                          hence should expose any address line connectivity issues.
@                          A sequence is written to memory, and then read back
@                          and checked against the expected sequence. Then,
@                          another sequence is written to memory, and this is continued
@                          until r3, the incrementer value, overflows. If
@                          the memory does not show any errors, it is assumed
@                          to be good. While this is not an exhaustive memory
@                          test, it is a simple test that can be used to verify
@                          basic memory functionality.
@
@ Arguments: r0 - starting address to test memory integrity
@            r1 - length of data to test memory integrity of
@
@ Return values: r0 - TRUE if success, FALSE if failure
@                r1 - value read from memory
@                r2 - value expected in memory
@                r3 - relative address where error occurred
@
@ Local variables: r0 - base address - unchanged
@                  r1 - length of memory - unchanged
@                  r2 - relative location into memory
@                  r3 - incrementer value used to generate sequences of data to load into memory
@                  r4 - value to load into memory
@
```

```
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: Does not verify that DRAM refresh is working correctly.
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     6/22/2017


.global mem_test
mem_test:
    PUSH {r4, r5, lr}
    LDR r3, =0              @Load incrementer value
    LDR r4, =0
mem_test_loop:
    LDR r5, =0x3F35D4B3
    ADDS r3, r5              @Increment incrementer value
    BCS success             @If have used all incrementer values, and thus overflowed, return
    LDR r2, =0              @Load the initial location to load into memory
    PUSH {r4}               @Store current starting value to recover later for when checking
writedata:
    STR r4, [r0, r2]        @Load value into memory
    ADDS r4, r3             @Increment value to load into memory
    SBC r4, r4, #1          @Subtract carry flag so that wrapping occurs at 2^32 + 1, not 2^32
    ADD r2, #4              @Increment the relative location to load into memory
    CMP r2, r1              @Check if written to all locations in memory
    BLT writedata           @If haven't, then keep writing, else go to check memory
    LDR r2, =0              @Reset initial location in memory
    POP {r4}                @Recover starting value and check data in memory
checkdata:
    LDR r5, [r0, r2]        @Load value from memory
    CMP r5, r4
    BNE failure
    ADDS r4, r3             @Increment value to load into memory
    SBC r4, r4, #1          @Subtract carry flag so that wrapping occurs at 2^32 + 1, not 2^32
    ADD r2, #4              @Increment the relative location to load into memory
    CMP r2, r1              @Check if written to all locations in memory
    BLT checkdata
    B mem_test_loop
failure:
    LDR r0, =FALSE
    MOV r1, r5
    MOV r3, r2
    MOV r2, r4
    B mem_test_end
success:
    LDR r0, =TRUE
    @B mem_test_end
```

```
mem_test_end:
    POP {r4, r5, pc}

.end
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                   pio.inc
@                       Contains definitions used by pio.s
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ Description: Defines register values used by the pio controller
@                   for this project
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017

@PIO Bank definitions
    .equ    PIO_A,      0           @Constant for specifying PIOA in config func
    .equ    PIO_B,      1           @Constant for specifying PIOB in config func
    .equ    PIO_C,      2           @Constant for specifying PIOA in config func

@PIO pin configuration definitions
@   Bit use for defining configurations:
@   Bit [1:0]   - PIO type (PIO_NORM, PIO_PerA, PIO_PerB)
@   Bit [2]     - PIO input or output (PIO_OUTPUT)
@   Bit [3]     - PIO interrupt enable (PIO_INT_EN)

    @ Bit [1:0]
    .equ    PIO_TYPE_MASK,  0x3     @Used to mask out PIO type config bits
    .equ    PIO_NORM,   0x0
    .equ    PIO_PERA,   0x1
    .equ    PIO_PERB,   0x2

    @ Bit [2]
    .equ    PIO_OUTPUT_MASK,    0x4 @Used to mask out PIO output config bits
    .equ    PIO_OUTPUT, 0x4

    @ Bit [3]
    .equ    PIO_INT_MASK,       0x8 @Used to mask out PIO interrupt config bits
    .equ    PIO_INT_EN, 0x8
```

```
@ Name              Comment              Date
@ Will Werst         Initial version     6/22/2017
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@                                    system.inc
@                        Contains general system-level definitions
@                            and definitions used by system.s
@
@
@
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ Description: Defines system-level constants, like TRUE or FALSE, and
@               also defines constants used by system.s
@
@ Revision History:
@ Name          Comment                  Date
@ Will Werst    Initial version          Some lonely night around 6/10/17
@ Will Werst    Comment                  October 2017

@ Memory definitions
    .equ    SRAM_START,     0x20000000
    .equ    SRAM_SIZE,      0x20000

    .equ    DRAM_START,     0x30000000
    .equ    DRAM_SIZE,      0x100000
    .equ    DRAM_NUM_ROWS,  1024
    .equ    DRAM_MILLIS_PER_REF, 256
    .equ    ROM_START,      0x80000000
    .equ    ROM_SIZE,       0x80000

@ Stack definitions

@ top of the stack, usually the end of the external SRAM
    .equ    TOP_STACK, 0x2001FFF0

@ interrupt stack size (see chapter 13 of the manual)
        .equ    IRQ_STACK_SIZE, 128

@ supervisor mode stack size (see chapter 13 of the manual)
    .equ    SVC_STACK_SIZE, 128


@ General Definitions
    .equ    TRUE,   1
    .equ    FALSE,  0


@ System function definitions
    @ timer definitions
    .equ    SLCK_CNT_SEC,   32768
    .equ    PIT_INTERVAL,   33          @SLCK ticks per PIT interrupt
    .equ    MILLIS_IN_SEC,  1000
```

```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@ System.s
@ Contains code for system-level functions, like timers
@
@
@ Description: Currently only contains code for initializing the system
@              clock.
@
@
@ Table of Contents:
@   - init_system: Call to initialize the timers in the system.
@   - elapsed_time: Returns the number of milliseconds since the last
@                   elapsed_time call.
@   - timerHandler: Handles the timer callbacks, and calling the oneMillisElapsed
@                   function when appropriate
@   - oneMillisElapsed: Called every time 1 ms elapses. This handles
@                       incrementing the millis shared variable, and
@                       anything else that the system needs, like
@                       DRAM refresh.
@
@
@
@
@
@
@ Revision History:
@ Name          Comment                 Date
@ Will Werst    Initial version         Some lonely night around 6/10/17
@ Will Werst    Comment                 October 2017



.include    "at91rm9200.inc"
.include    "macro.inc"
.include    "system.inc"


.text
.arm

@ init_system
@
@ Description: Initializes the code in this file. Call before
@              using anything in this file.
@
@ Operational Description: A timer interrupt is setup, and then
@                          some status registers are read to clear them.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
```

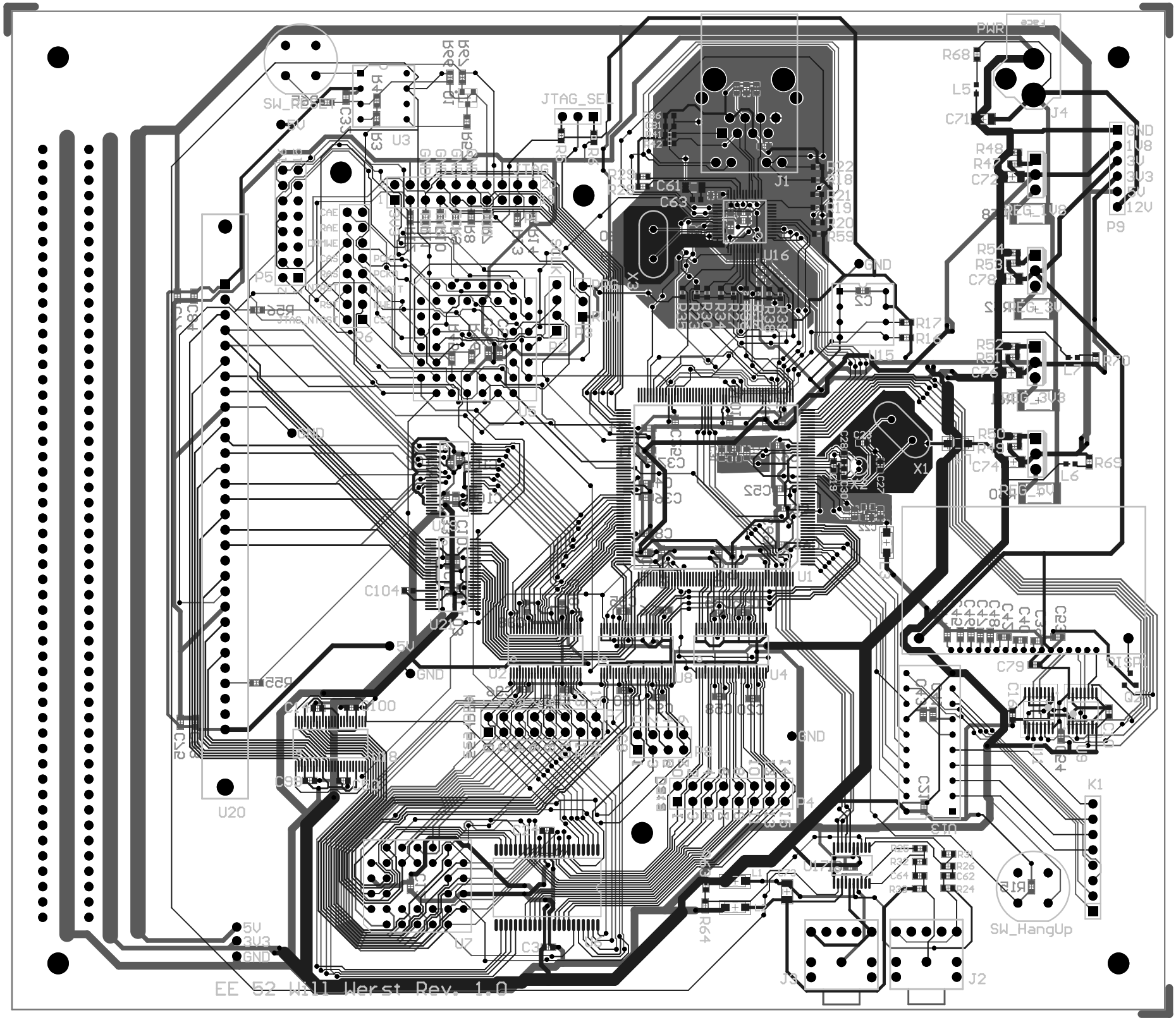
```
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global init_system
init_system:
    mSTARTFNC
    mSET_HREG   AIC_SMR1,   0x00000021      @Initialize the system timer interrupt
    mSET_HREG AIC_SVR1, timerHandler        @Set callback
    mSET_HREG   ST_PIMR,    PIT_INTERVAL    @Set timer period
    mSET_HREG   ST_IER,     0x00000001      @Enable interrupt
    mSET_HREG AIC_IECR,     0x00000002      @Enable interrupt
    LDR     r0,     =ST_SR                  @Read status registers to clear them
    LDR     r0,     [r0]
    LDR     r0,     =RTC_SR                 @Read status registers to clear them
    LDR     r0,     [r0]
    LDR     r0,     =PMC_SR                 @Read status registers to clear them
    LDR     r0,     [r0]
    mRETURNFNC


@ elapsed_time
@
@ Description: Returns the number of milliseconds which have elapsed since the
@              last time this function was called.
@
@ Operational Description: The current millis count is loaded, and
@                          the last_millis count is loaded as well.
@                          The result is generated by subtracting last_millis
@                          from millis, and then last_millis is set to the current
@                          value of millis.
@
@ Arguments: None
@
@ Return values: r0 - number of milliseconds since the last time the function
@                     was called.
@
@ Local variables: None
@
@ Shared variables: millis[R] - the current millis count is read
@                   last_millis[RW] -  the previous millis count is read and updated
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
```

```
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None known
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

@ returns: number of milliseconds since last function call
.global elapsed_time
elapsed_time:
    mSTARTFNC
    mLoadToReg  r0, millis              @load current milliseconds
    mLoadToReg  r1, last_millis         @load last milliseconds
    mStoreFromReg   r0, r2, last_millis @store current milliseconds into last milliseconds
    SUB     r0, r0, r1                  @get difference

    mRETURNFNC


@ timerHandler
@
@ Description: Handles incrementing the timer_counter.
@              This function is called from an interrupt.
@
@ Operational Description: The timer_counter variable is
@                   incremented by the number of milli-ticks of slow clock
@                   per interrupt, and then any full milliseconds are
@                   removed from timer_counter and the oneMillisElapsed
@                   function is called for each one of these full
@                   milliseconds.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: timer_counter[RW] - This counter
@                       contains the number of milli-ticks
@                       of the slow clock. It is updated by method.
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
```

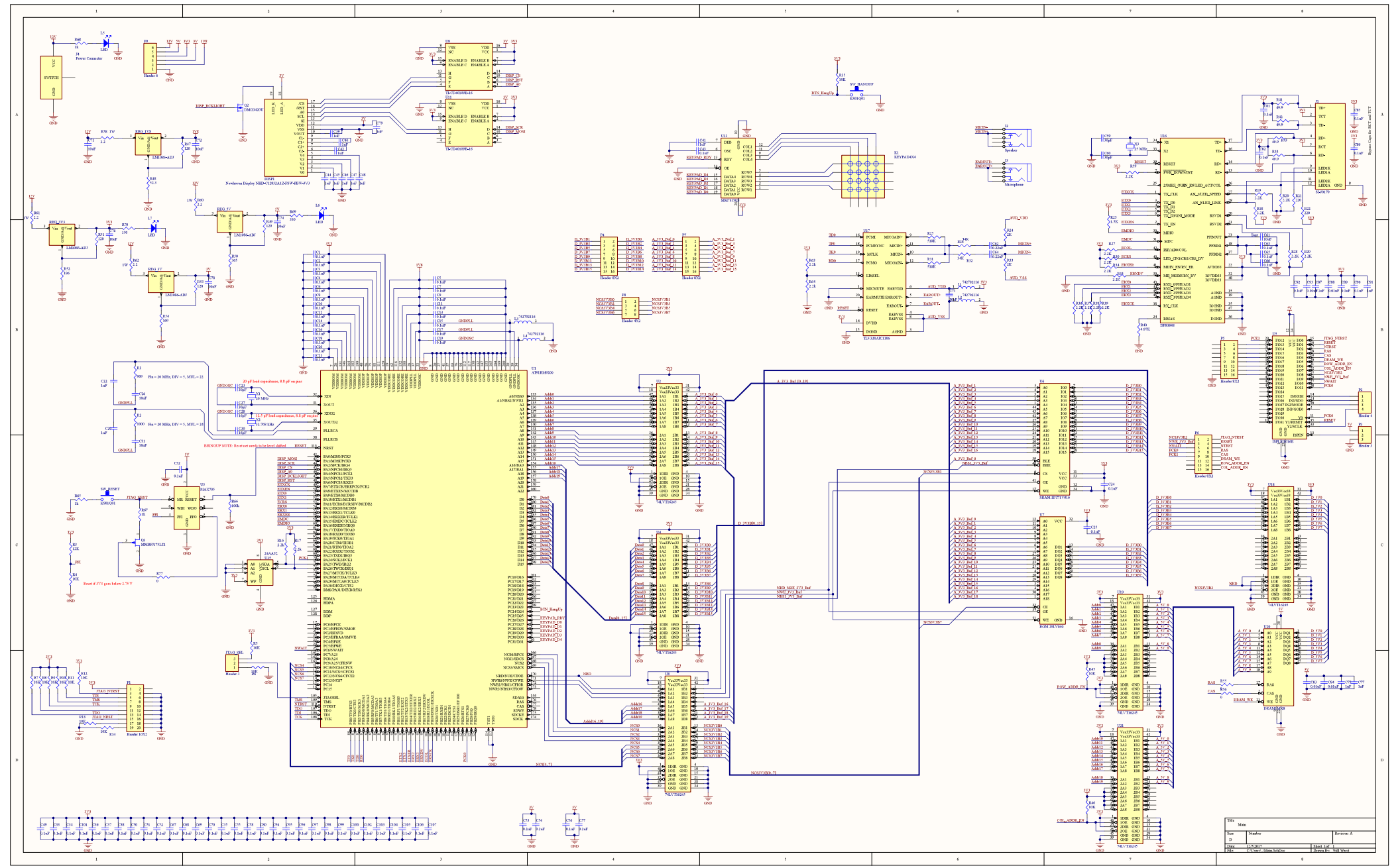
```
@ Revision History:
@ Name              Comment              Date
@ Will Werst         Initial version     Some lonely night around 6/10/17

@ variables:
@   r1 - timer_counter
timerHandler:
    mSTARTINT
    LDR     r0,     =ST_SR              @Read status register to clear it
    LDR     r0,     [r0]
    LDR     r0,     =RTC_SR             @Read RTC register to clear it
    LDR     r0,     [r0]
    LDR     r0,     =PMC_SR             @Read power management controller
    LDR     r0,     [r0]                @status register to clear it
    LDR     r0,     =(PIT_INTERVAL*MILLIS_IN_SEC) @Load number of milli-ticks
                                        @of slow clock between interrupts.
    mLoadToReg  r1, timer_counter       @And add the number of milli-ticks
    ADD     r1, r1, r0                  @to the current count of milli-ticks
transferFullMillisCounts:
    CMP     r1, #SLCK_CNT_SEC           @Check to see if a millisecond has elapsed
    BLO     endTimerHandler             @No full milliseconds to transfer
    PUSH    {r1}
    BL      oneMillisElapsed            @Call the function to increment millis
                                        @and also take any action that should
                                        @be done when a millisecond has elapsed
    POP     {r1}
    SUB     r1, #SLCK_CNT_SEC           @Take off the number of counts in timer_counter
                                        @associated with one millisecond elapsing
    B       transferFullMillisCounts    @Go back and check if more milliseconds to transfer
endTimerHandler:
    mStoreFromReg   r1, r0, timer_counter @Save the remainder milli-ticks
    mRETURNINT


@ oneMillisElapsed
@
@ Description: This function should be called by the timerHandler
@              every time one millisecond elapses.
@
@ Operational Description: The millis variable is incremented, and then
@                          any other actions that should be taken every
@                          millisecond are done. Currently this is only
@                          a DRAM refresh.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
```

```
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17


@ oneMillisElapsed
@ Description: Function used to increment millisecond, and also do any other
@               actions such as refresh DRAM
@

oneMillisElapsed:
    mSTARTFNC
    mLoadToReg  r0, millis          @Load millis
    ADD     r0, #1                  @Increment millis
    mStoreFromReg   r0, r2, millis  @save millis

    @Refresh DRAM, need to refresh 1024 rows every 16 ms
    LDR     r0,     =((DRAM_NUM_ROWS / DRAM_MILLIS_PER_REF)) @Figure out how many rows to refresh
    LDR     r1,     =DRAM_START     @Load start address of DRAM
    mLoadToReg  r2,     dramRefreshRow  @Load the row that was refreshed last
dramRefresh:
    LDRB    r3, [r1, r2]            @Refresh row
    ADD     r2,     #1              @Go to next row
    CMP     r2,     #DRAM_NUM_ROWS  @Check if gone through all rows
    LDRHS   r2,     =0              @If so, go back to row 0
    SUBS    r0,     #1              @Decrement row counter
    BHI     dramRefresh             @If still rows to refresh, continue refreshing
    @B      endDRAMRefresh          @Done refreshing

endDRAMRefresh:
    mStoreFromReg   r2, r0, dramRefreshRow @Save row that was worked on
    @B  endOneMillisElapsed
endOneMillisElapsed:
    mRETURNFNC                      @Done, return



.data

millis:                             @Stores milliseconds elapsed since start
    .word 0x00000000
last_millis:                        @Stores milliseconds since last elapsed_time call
    .word 0x00000000
timer_counter:                      @Counter for converting slow clock ticks to milliseconds
    .word 0x00000000
dramRefreshRow:                     @Counter for which row to refresh next
    .word 0x00000000

.end
```

Appendix F (Printed Circuit Board)


JTAG_SEL
SW_HangUp
P9
P5
P4
J1
J2
J3
J4
K1
X1
U1
U2
U3
U4
U5
U7
U8
U15
U16
U20
PWR
GND
5V
3V3
EE 52 Will Werst Rev. 1.0

Appendix G (Schematics)


ATMEGA2560
Newhaven Display NHD-C12832A1Z-FSW-FBW-3V3
LM1086-ADJ
LM1085-ADJ
MAX705
24AA32
MMBFJ175LT1
IDT74FCT3245
IDT74FCT162245
IDT70V3599
IDT71V016
IDT71V67803
IDT71V67903
IDT71V3578
IDT71V3579
IDT71V3577
IDT71V3576
IDT71V3557
IDT71V3558
IDT71V3556
IDT71V3546
IDT71V3547
IDT71V3548
IDT71V3549
IDT71V3568
IDT71V3569
IDT71V3570
IDT71V3571
IDT71V3572
IDT71V3573
IDT71V3574
IDT71V3575
IDT71V3589
IDT71V3590
IDT71V3591
IDT71V3592
IDT71V3593
IDT71V3594
IDT71V3595
IDT71V3596
IDT71V3597
IDT71V3598
IDT71V3599
IDT71V3560
IDT71V3561
IDT71V3562
IDT71V3563
IDT71V3564
IDT71V3565
IDT71V3566
IDT71V3567
IDT71V3580
IDT71V3581
IDT71V3582
IDT71V3583
IDT71V3584
IDT71V3585
IDT71V3586
IDT71V3587
IDT71V3588
IDT71V3550
IDT71V3551
IDT71V3552
IDT71V3553
IDT71V3554
IDT71V3555
IDT71V3540
IDT71V3541
IDT71V3542
IDT71V3543
IDT71V3544
IDT71V3545
IDT71V3530
IDT71V3531
IDT71V3532
IDT71V3533
IDT71V3534
IDT71V3535
IDT71V3536
IDT71V3537
IDT71V3538
IDT71V3539
IDT71V3520
IDT71V3521
IDT71V3522
IDT71V3523
IDT71V3524
IDT71V3525
IDT71V3526
IDT71V3527
IDT71V3528
IDT71V3529
IDT71V3510
IDT71V3511
IDT71V3512
IDT71V3513
IDT71V3514
IDT71V3515
IDT71V3516
IDT71V3517
IDT71V3518
IDT71V3519
IDT71V3500
IDT71V3501
IDT71V3502
IDT71V3503
IDT71V3504
IDT71V3505
IDT71V3506
IDT71V3507
IDT71V3508
IDT71V3509