CMPE-013/L

Introduction to "C" Programming

Max Lichtenstein

Roadmap

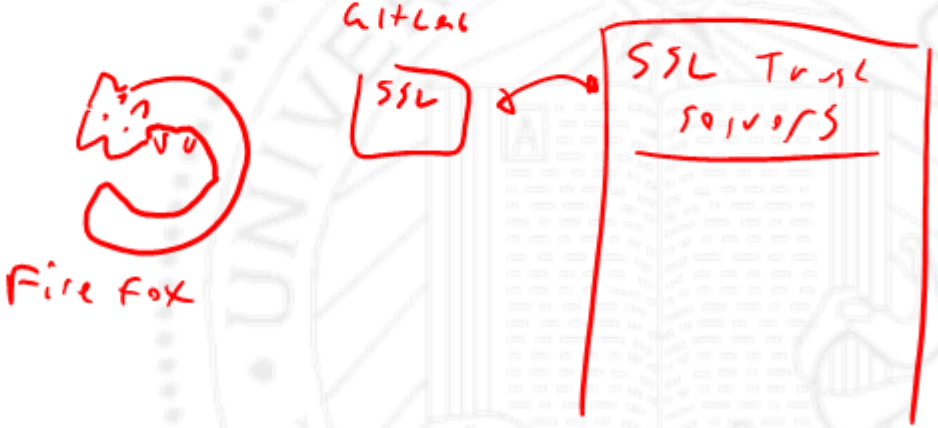
- Announcements
- ~~Grades Review~~
- Timer Question
- Battleboats
  - Teamwork expectations
  - Overview
    - Demo
- BREAK
  - Negotiation Module
  - Transmission Module
  - Message Module
  - Field Module
  - Agent SM Module
- BREAK (?)
  - **Parallel SMs (?)**
  - **Transmission protocols (?)**
  - **Randomness (?)**
  - **Hashing (?)**

Announcements

- Lab 8 extended to tonight at midnight
- Lab 9 postponed: Out today, due next Tuesday
- Lab 10 proposal deadline extended to Tomorrow

Announcements

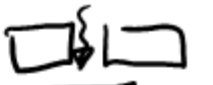
- Gitlab still “down”!
  - See piazza post on workaround
  - If HTTPS isn’t back up by Thursday, this workaround will be required!

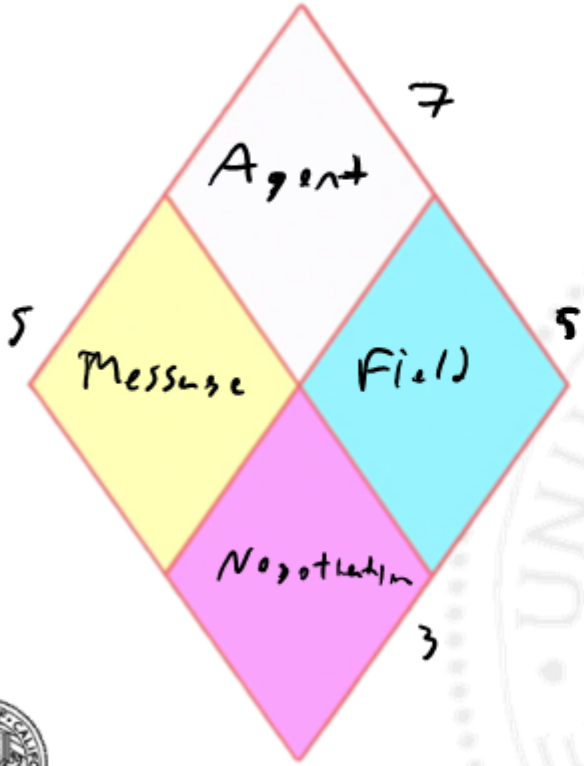


Max Lichtenstein

Battleboats:
TEAMWORK EXPECTATIONS

Battleboats: Teamwork

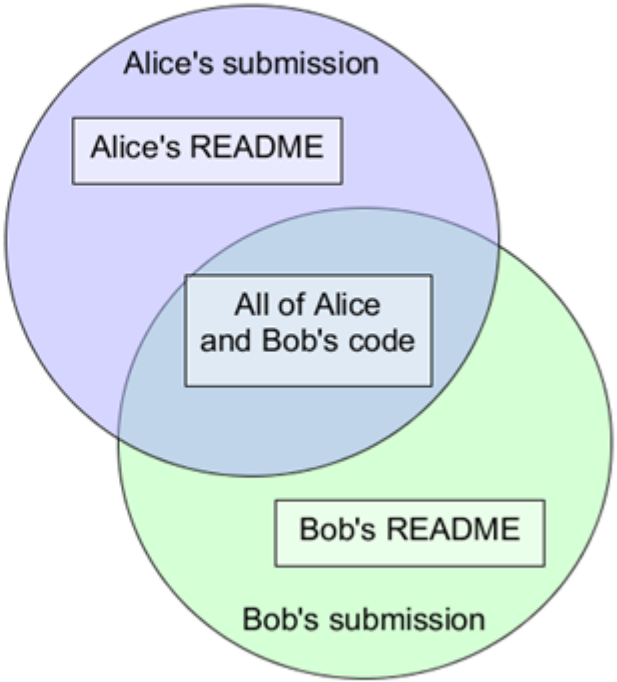




- 4 Modules to write
  - Each partner writes 2 modules
  - And the other 2 test harnesses
  - You pick
- Both partners must:
  - understand each other's code

Battleboats: Partnered Submission

- Both partners are graded on the same code
  - Only one partner has to git submit the code base
- But each turns in their own README.txt
  - Both partners git submit a README



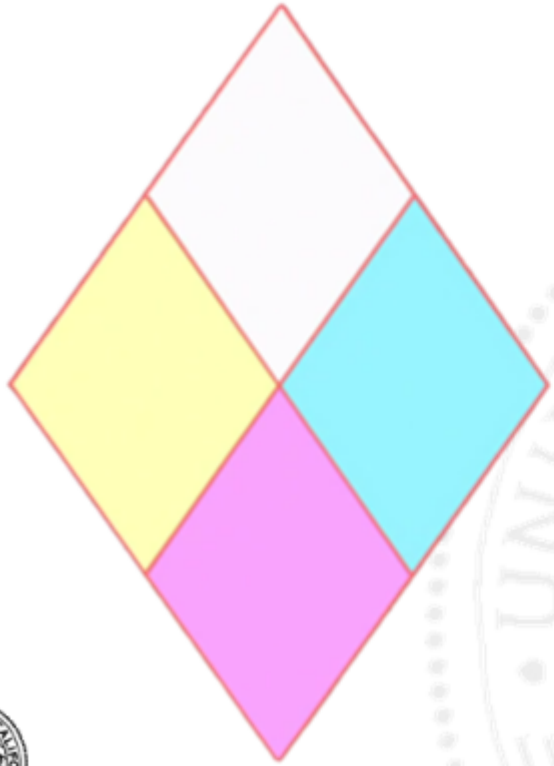
Battleboats: Partnered READMEs

CruzID: alice@ucsc.edu	CruzID: bob@ucsc.edu
MY REPO CONTAINS THE FINAL VERSION OF THE PROJECT.	ALICE'S REPO CONTAINS THE FINAL VERSION OF THE PROJECT.
I WROTE: -AGENT.C -NEGOTIATION.C. -MESSAGETEST.C -FIELDTEST.C	I WROTE: -AGENTTEST.C -NEGOTIATIONTEST.C. -MESSAGE.C -FIELD.C
Please note we implemented the Field AI for extra credit, and it beats the staff AI 95% of the time.	Our AI destroys yours, give me extra points for that.
Introduction: In this lab, we...	Introduction: Lab09 was absolutely wild. I ...

← on Top

Max Lichtenstein

Battleboats: Solo option



- Pick 2 options
  - If you pick Negotiation you must also do an extra test harness
- You still must:
  - understand the whole system
- Y'all get .9

Battleboats: Solo READMEs

CruzID: carol@ucsc.edu

صاوک

I WROTE:

-AGENT.C

-NEGOTIATION.C.

-AGENT_TEST.C

-NEGOTIATION_TEST.C

-FIELDTEST.C

I implemented a human agent for extra credit.

Introduction:

This lab was so unbelievably ....

Battleboats: How to work as a team

- Communicate frequently
  - Schedule ✓
  - Workload ✓
  - Strategy ✓
  - Comprehension ✓
- Trust is required ✓
  - Test harnesses can build trust!

Battleboats: Team expectations

- Conflict resolution
  - Be professional adults
  - Listen to each other
  - Respect each other
  - Tell staff EARLY if there are serious issues!
- Work Together
  - Time
  - Don't destroy each other's work
    - (unless you agree to)
- Share workload

Battleboats: Overview

BattleBoats v2

- More modular
  - Easier in some ways!
    - More visibility, Easier to debug
    - Easier to divide and conquer
  - harder in others:
    - Top level is a bit harder to understand
    - More diagrams are required

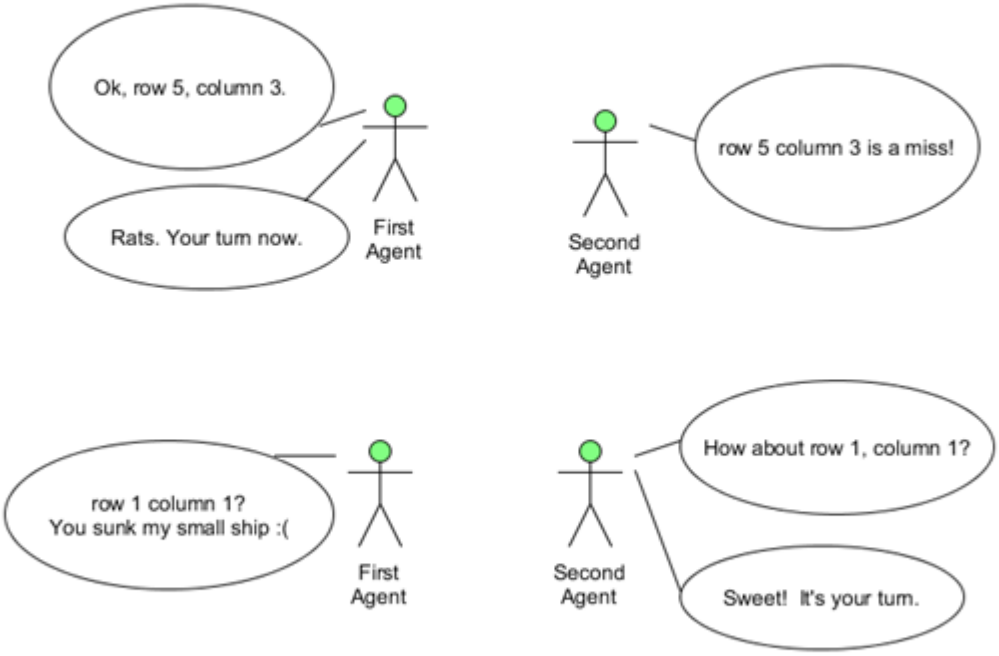
BattleBoats v2

- New Who-goes-first protocol
  - Uses roles-based handshake
  - More intuitive (maybe?)
  - More work (sort of?)

What is Battleboats?

What is Battleboats?

What is Battleboats?



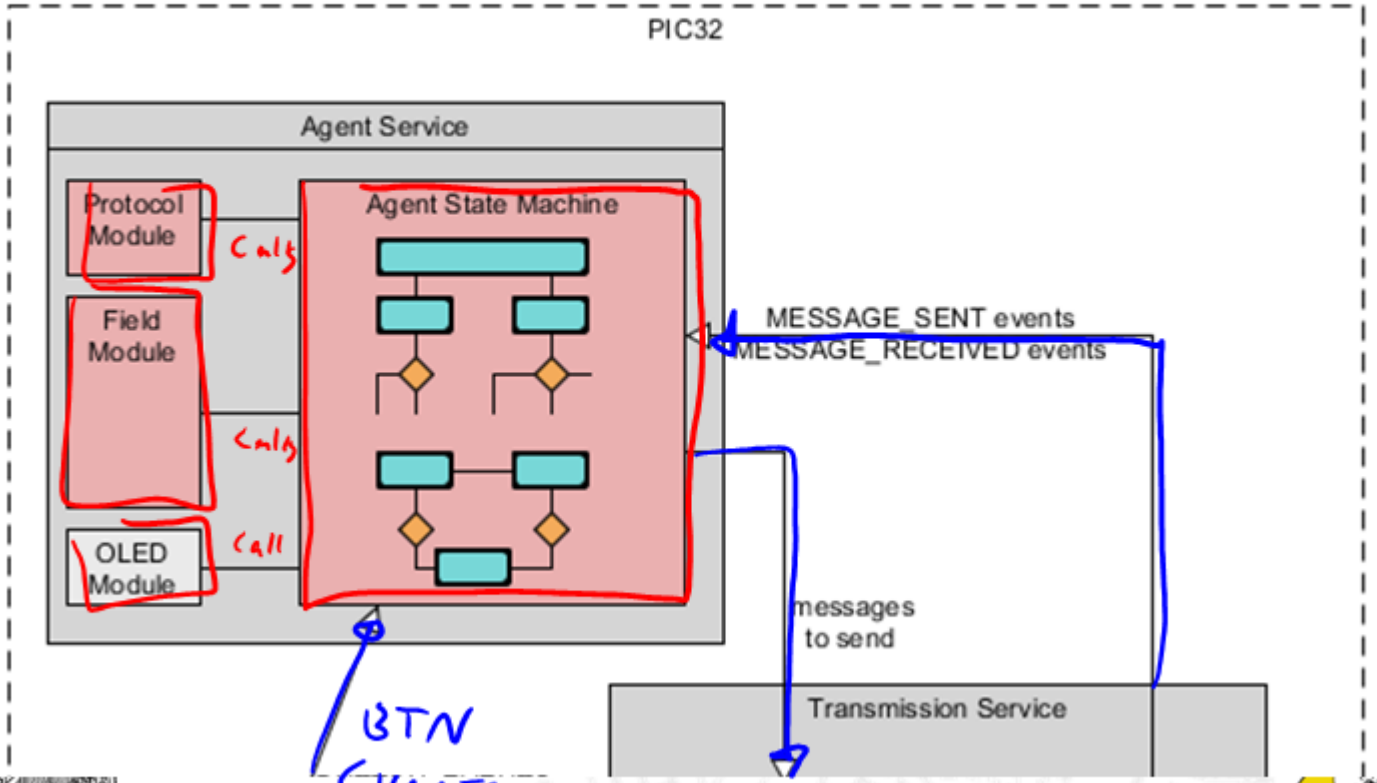
What is Battleboats?

- 2 Player game
- Each player keeps 2 grids:
  - Their own
  - Estimate of their opponent's
- Take turns guessing squares
  - You lose if all your boats sink
- A game of chance?
  - Not entirely!

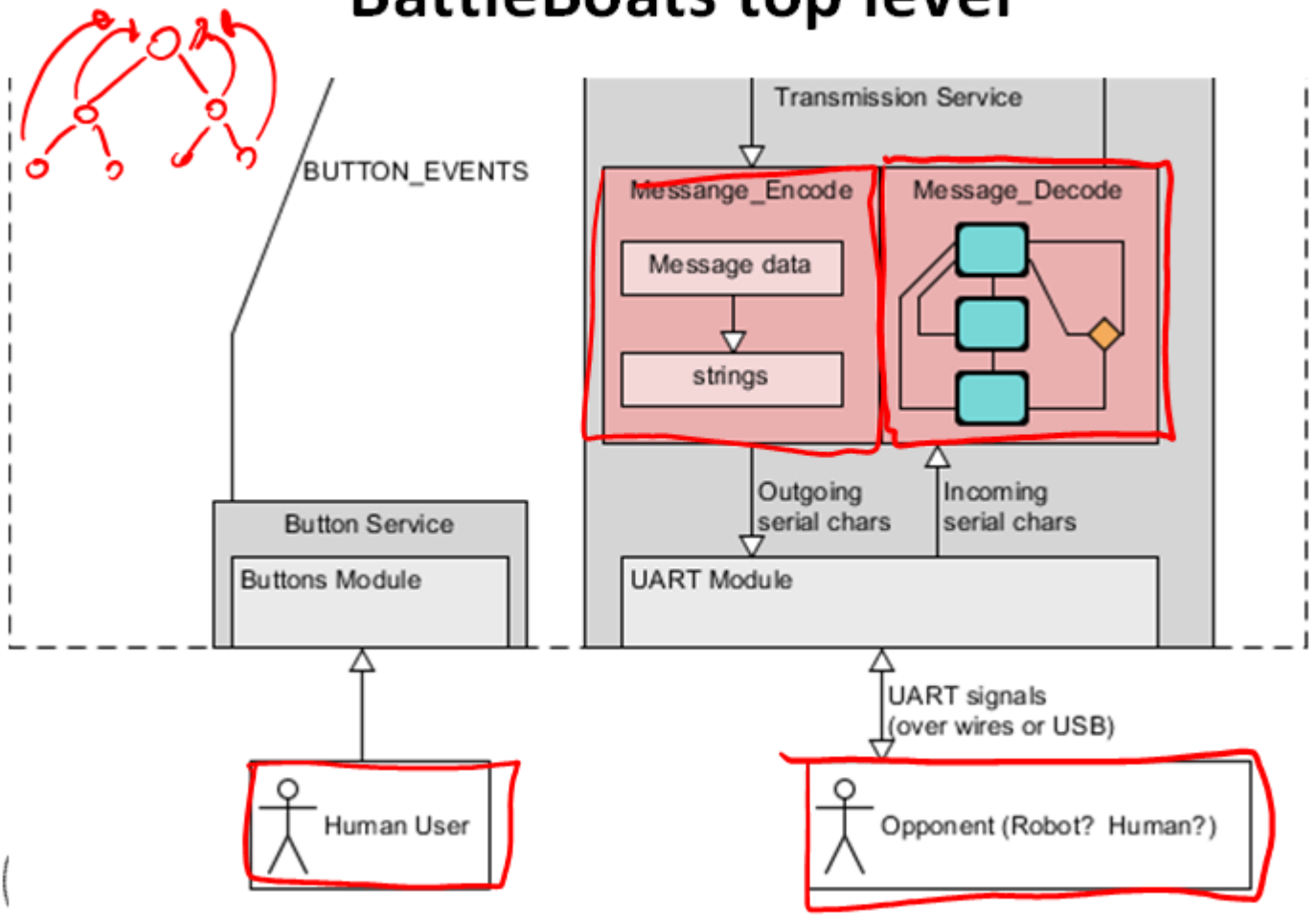
What is Battleboats, *Really*?

- Like any turn-based game:
  - A state machine (or pair of parallel SMs)
  - A communication protocol
  - Two intelligent agents
    - Or at least, 2 agents which can correctly follow the rules

BattleBoats top level

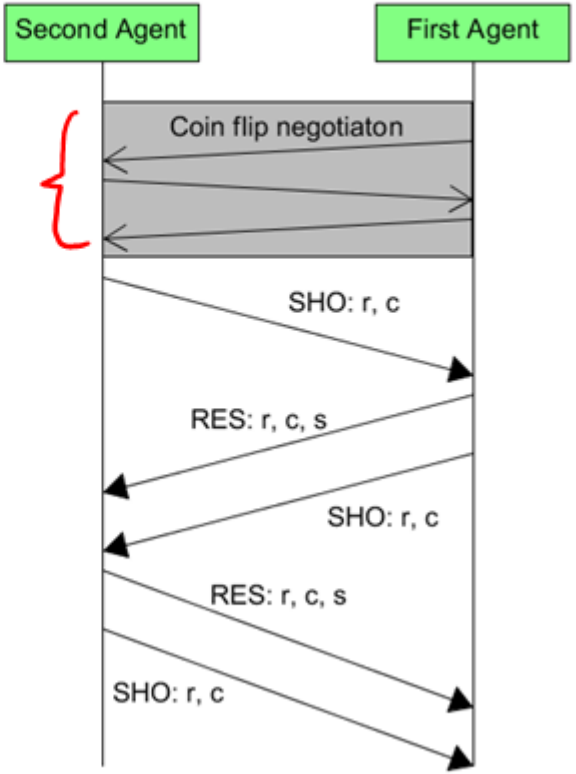


BattleBoats top level



Battleboats gameplay

- Coin flip
- followed by turn taking



Battleboats for Computers:

- Cryptographic "coin flip" protocol
  - A "commitment scheme"
- Specialized message format
  - `$RES,2,2,0*58`
  - `$SHO,4,11*60`
- Randomness

READ IT!

Main file:

Main file:

Main file:

```c
int main()
{
    BOARD_Init();
    Uart1Init(UART_BAUD_RATE);
    OpenTimer2(T2_ON | T2_SOURCE_INT | T2_PS_1_16, BOARD_GetPBClock() / 16 / 100);
    ButtonsInit();
    OledInit();
    AgentInit();

    //Main loop:
    while (TRUE) {
        if (battleboatEvent.type != BB_EVENT_NO_EVENT) {
        AgentRun(battleboatEvent);
        if (message_to_send.type != MESSAGE_NONE) {
            Transmission_StartSendingMessage(&message_to_send);
        }
        battleboatEvent.type = BB_EVENT_NO_EVENT;
        }
    }
}

void __ISR(_TIMER_2_VECTOR, IPL4AUTO) Timer1Interrupt100Hz(void)
{
    uint8_t buttonEvents = ButtonsCheckEvents();
    if (buttonEvents & BUTTON_EVENT_4UP) {
        battleboatEvent.type = BB_EVENT_START_BUTTON;
    }

    Transmission_SendChar();
    Transmission_ReceiveChar();
}
```

Not ACCURATE

if evt

AGENTSM

send mess

Consume Event

✓ buttons

sending server

wfor messages

Demo!

Then BREAK

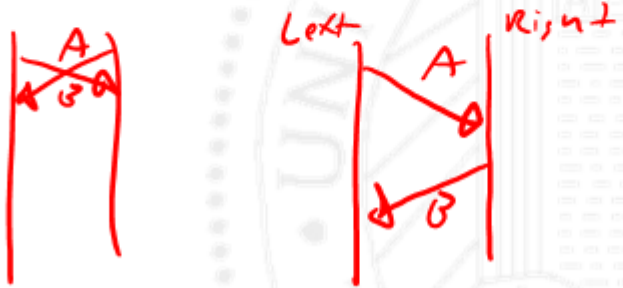
Negotiation

Who goes first

- Cannot flip a coin!
  - Why not generate a random number?

DICE COULD LIE

  - Why not have both parties generate random numbers?



Coin Flip Commitment Scheme

1. Alice generates a secret (random) number, A
2. Alice uses her secret number to generates a commitment number, #a
   - She uses a hash (a one-way function)
   - She sends it to Bob
3. Bob generates a (random) number, B
   - He sends it to Alice
4. Alice sends A to Bob
5. Alice and Bob can both use A and B to determine heads or tails
6. And, Bob can compare #a and A to ensure Alice kept her commitment

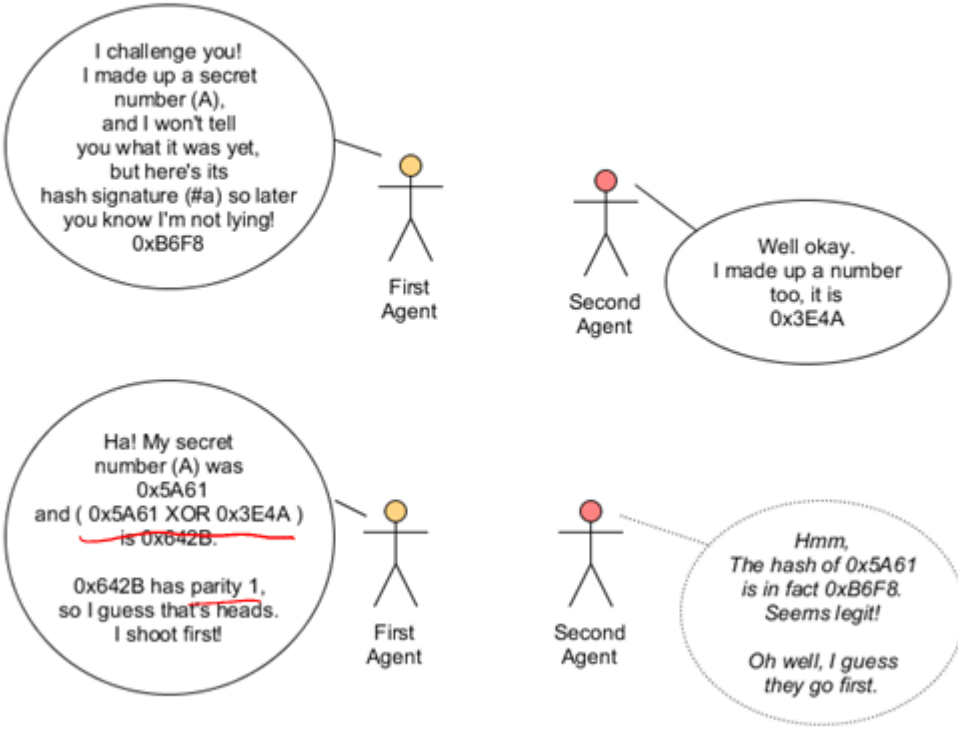
$a\# = hash(A)$

$A = hash^{-1}(a\#)$

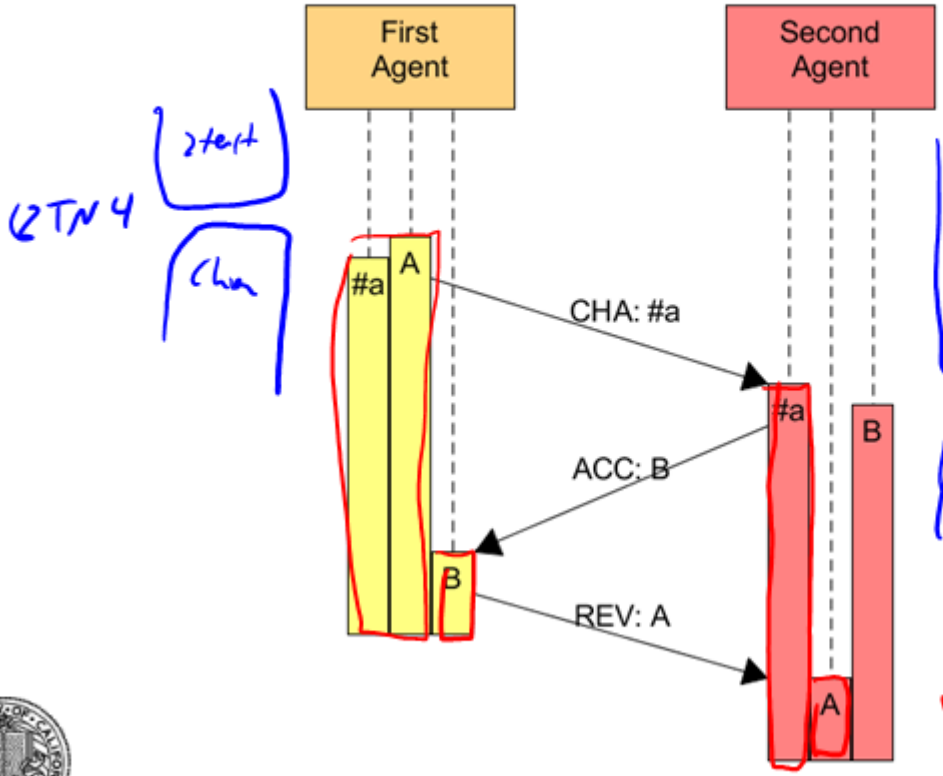
$a = A \oplus A$

$A = \sqrt{a}$

Commitment Scheme



Commitment Scheme



Commitment Scheme Details

- One way function is “beef hash”:

  #a = A^2 mod 0xBEEF

- CoinFlip(A,B) = parity(A XOR B)

0 1 1 0 1 0 0 0
1 2 3

3 is odd

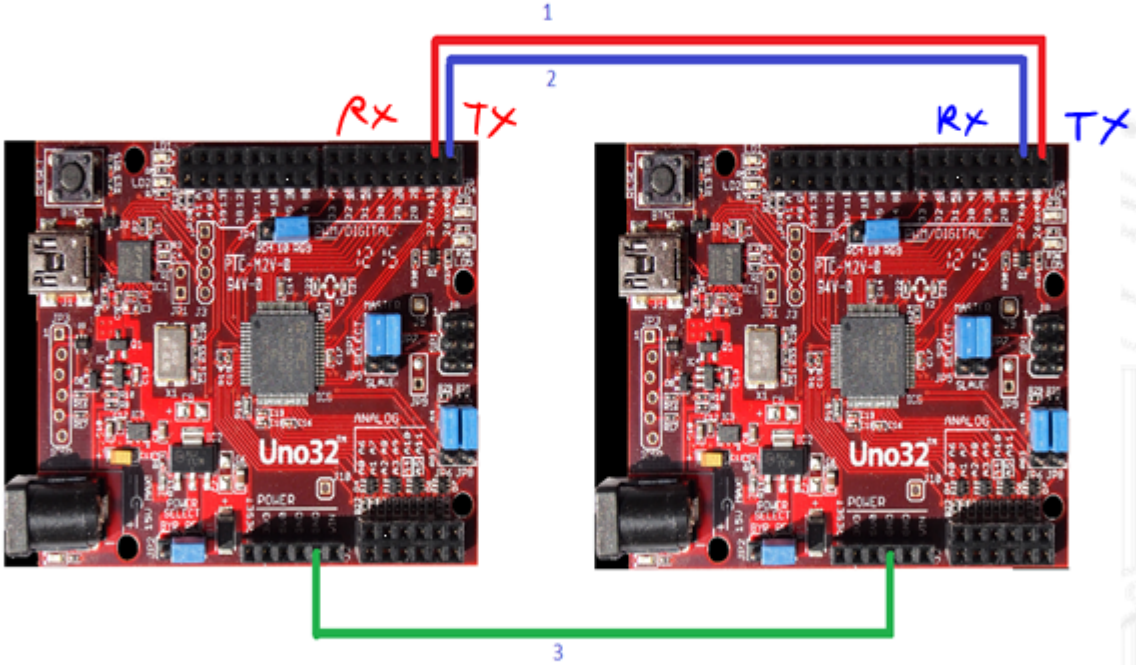
so parity = 1

HEADS

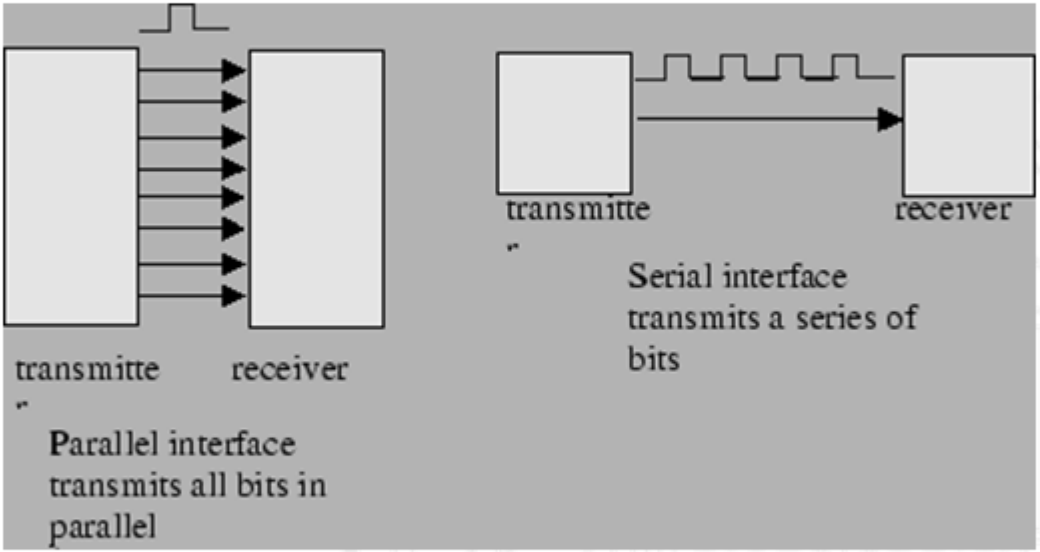
Max Lichtenstein

Transmission

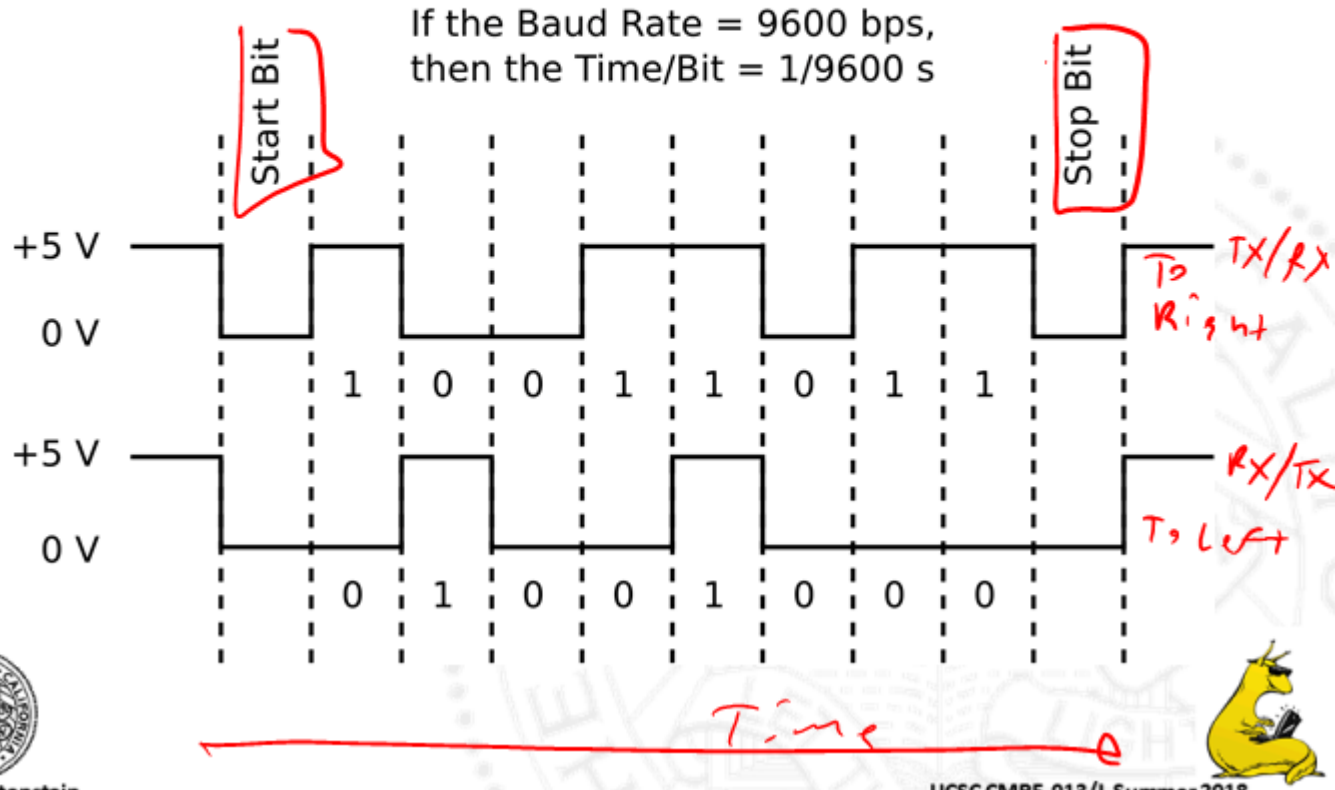
Serial Communication (UART)



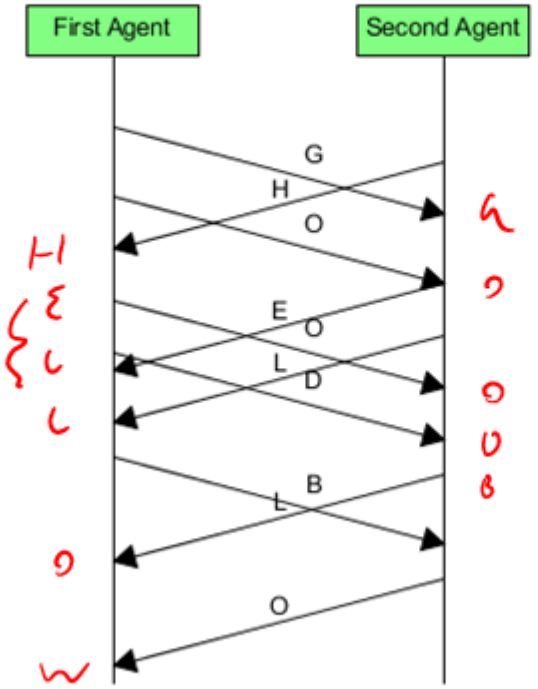
Serial Communication (UART)



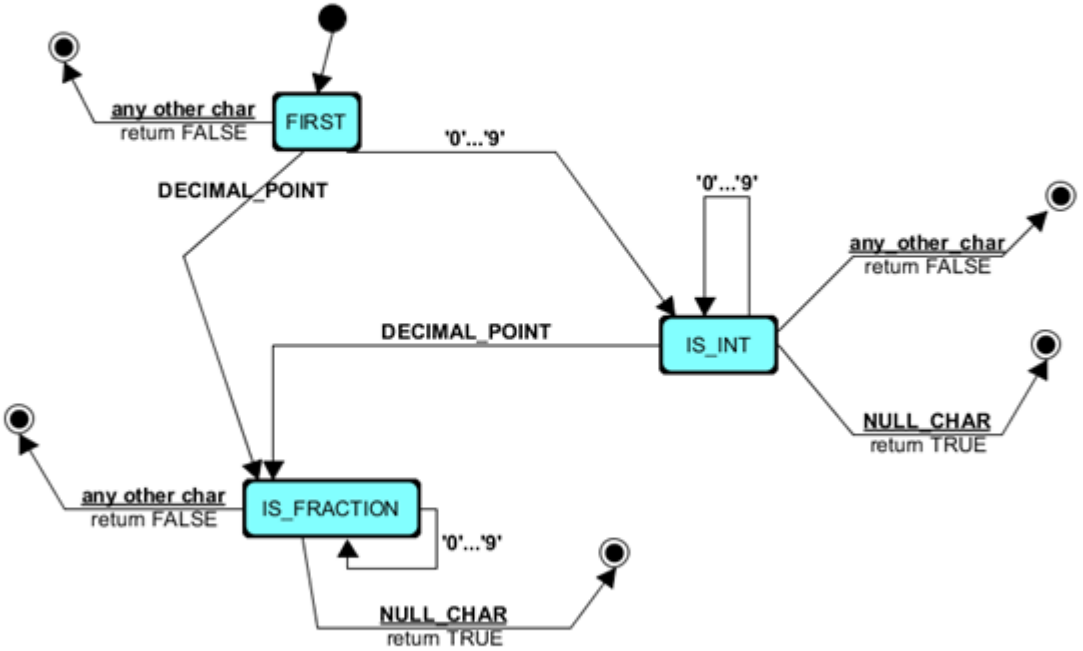
Serial Communication (UART)



Serial Communication

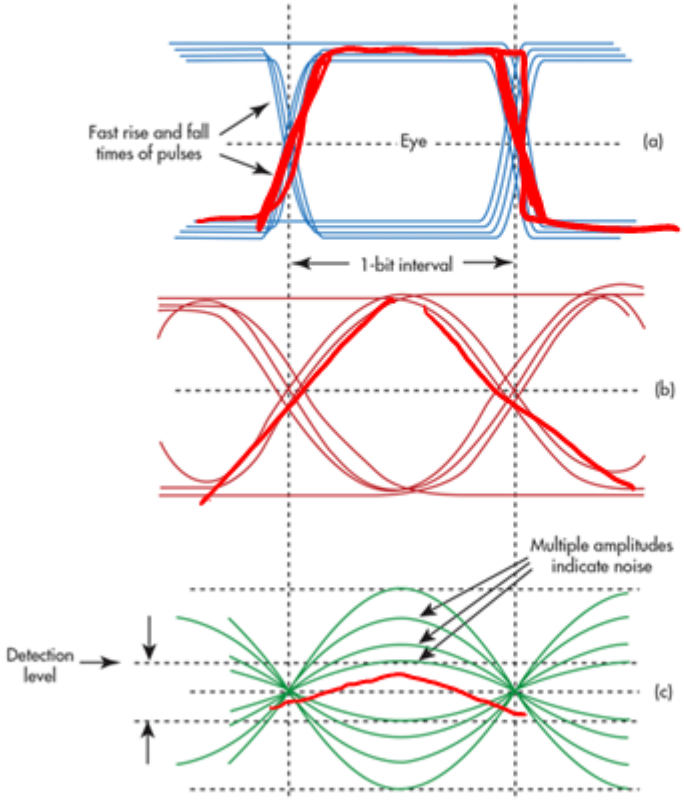


Serial Communication



Message

Bit flipping



1 0 ← oops

0 0

0 0

1 1

0 0

1 1

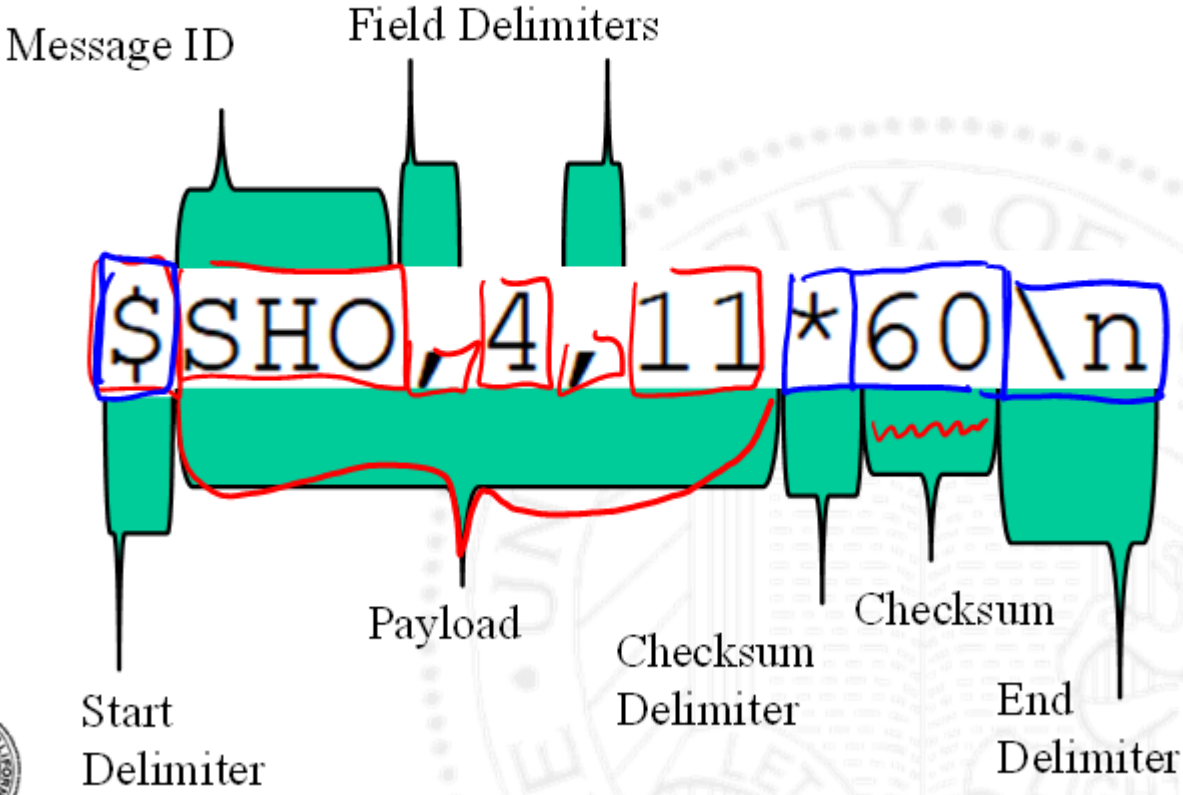
1 1

0 0

NMEA ~~NEMA~~ Message Protocol

```
$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,*76
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,19,13,28,070,17,26,23,252,,04,14,186,14*79
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092750.000,A,5321.6802,N,00630.3372,W,0.02,31.66,280511,,,A*43
$GPGGA,092751.000,5321.6802,N,00630.3371,W,1,8,1.03,61.7,M,55.3,M,,*75
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,16,13,28,070,17,26,23,252,,04,14,186,15*77
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092751.000,A,5321.6802,N,00630.3371,W,0.06,31.66,280511,,,A*45
```

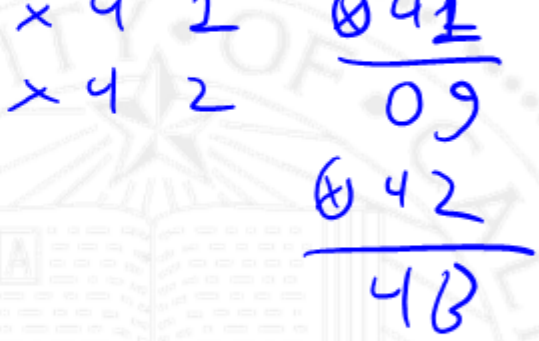
NMEA Message Protocol



Checksums

- XOR each char together

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@
33	21	041	!	!	65	41	101	A	A
34	22	042	"	"	66	42	102	B	B
35	23	043	#	#	67	43	103	C	C
36	24	044	$	$	68	44	104	D	D
37	25	045	%	%	69	45	105	E	E
38	26	046	&	&	70	46	106	F	F
39	27	047	'	'	71	47	107	G	G
40	28	050	(	(	72	48	110	H	H
41	29	051	)	)	73	49	111	I	I
42	2A	052	*	*	74	4A	112	J	J
43	2B	053	+	+	75	4B	113	K	K
44	2C	054	,	,	76	4C	114	L	L
45	2D	055	-	-	77	4D	115	M	M
46	2E	056	.	.	78	4E	116	N	N
47	2F	057	/	/	79	4F	117	O	O
48	30	060	0	0	80	50	120	P	P
49	31	061	1	1	81	51	121	Q	Q
50	32	062	2	2	82	52	122	R	R
51	33	063	3	3	83	53	123	S	S
52	34	064	4	4	84	54	124	T	T
53	35	065	5	5	85	55	125	U	U
54	36	066	6	6	86	56	126	V	V
55	37	067	7	7	87	57	127	W	W
56	38	070	8	8	88	58	130	X	X
57	39	071	9	9	89	59	131	Y	Y
58	3A	072	:	:	90	5A	132	Z	Z
59	3B	073	;	;	91	5B	133	[	[
60	3C	074	<	<	92	5C	134	\	\
61	3D	075	=	=	93	5D	135	]	]
62	3E	076	>	>	94	5E	136	^	^
63	3F	077	?	?	95	5F	137	_	_



Checksums

- XOR each char together

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@
33	21	041	!	!	65	41	101	A	A
34	22	042	"	"	66	42	102	B	B
35	23	043	#	#	67	43	103	C	C
36	24	044	$	$	68	44	104	D	D
37	25	045	%	%	69	45	105	E	E
38	26	046	&	&	70	46	106	F	F
39	27	047	'	'	71	47	107	G	G
40	28	050	(	(	72	48	110	H	H
41	29	051	)	)	73	49	111	I	I
42	2A	052	*	*	74	4A	112	J	J
43	2B	053	+	+	75	4B	113	K	K
44	2C	054	,	,	76	4C	114	L	L
45	2D	055	-	-	77	4D	115	M	M
46	2E	056	.	.	78	4E	116	N	N
47	2F	057	/	/	79	4F	117	O	O
48	30	060	0	0	80	50	120	P	P
49	31	061	1	1	81	51	121	Q	Q
50	32	062	2	2	82	52	122	R	R
51	33	063	3	3	83	53	123	S	S
52	34	064	4	4	84	54	124	T	T
53	35	065	5	5	85	55	125	U	U
54	36	066	6	6	86	56	126	V	V
55	37	067	7	7	87	57	127	W	W
56	38	070	8	8	88	58	130	X	X
57	39	071	9	9	89	59	131	Y	Y
58	3A	072	:	:	90	5A	132	Z	Z
59	3B	073	;	;	91	5B	133	[	[
60	3C	074	<	<	92	5C	134	\	\
61	3D	075	=	=	93	5D	135	]	]
62	3E	076	>	>	94	5E	136	^	^
63	3F	077	?	?	95	5F	137	_	_

Field

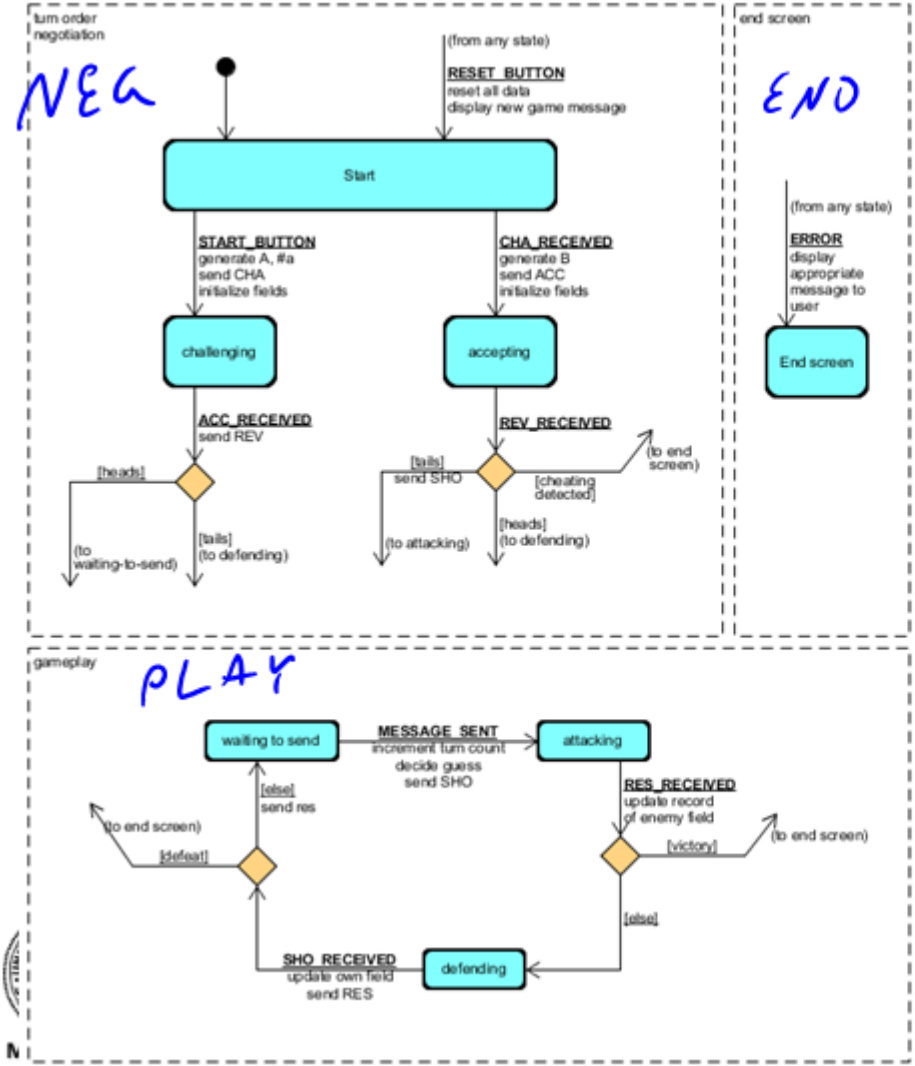
Field

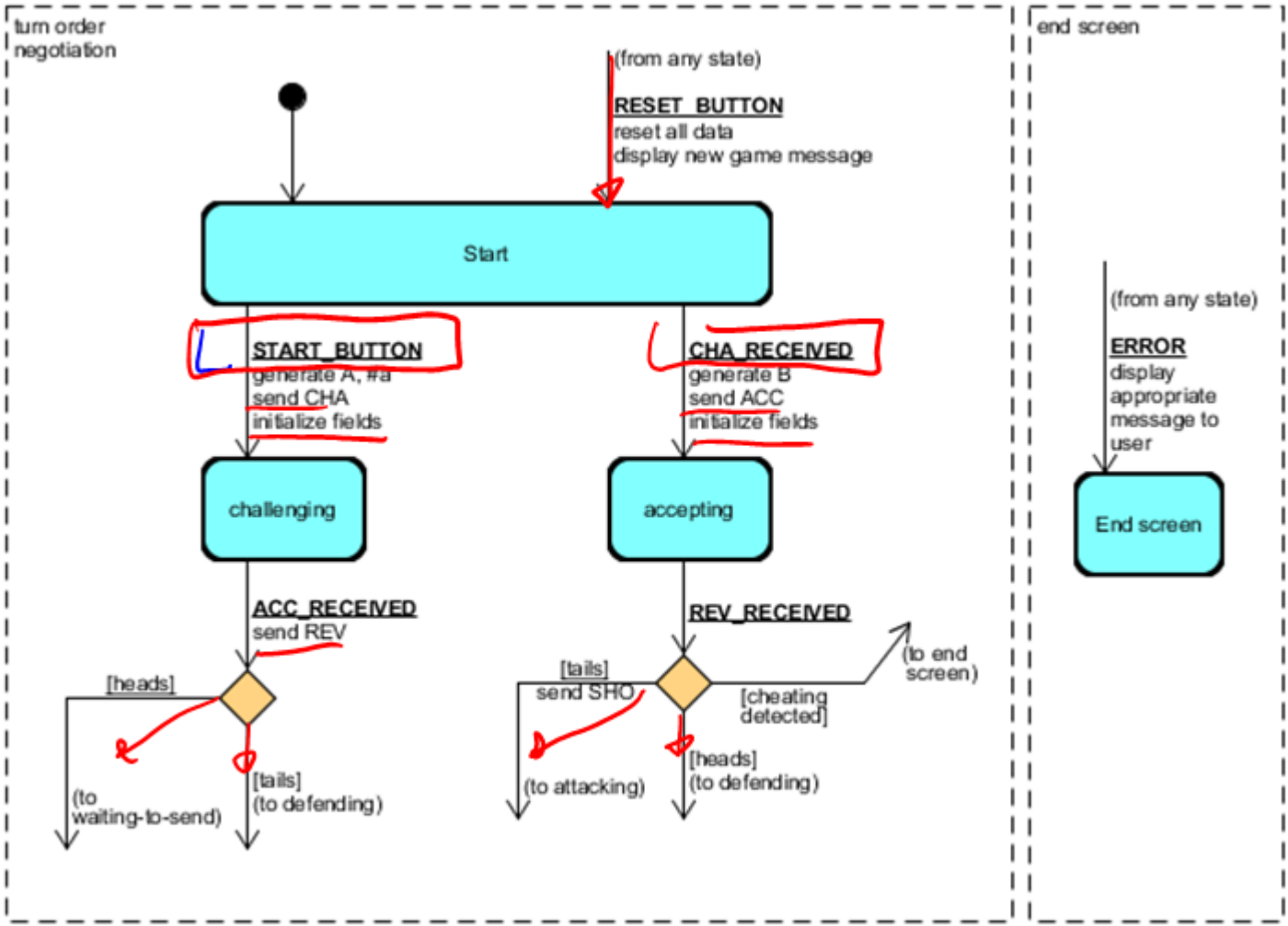
Agent

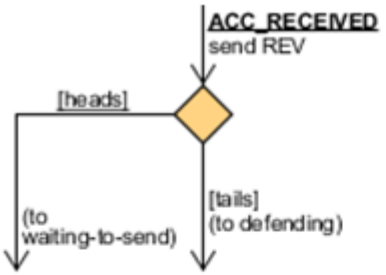
Agent

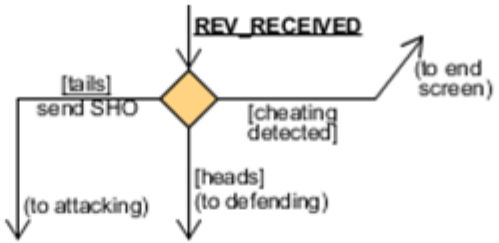
- Responsible for:
  - Ordering Communication
  - Communicating with user
  - Connecting the other modules to create gameplay

Agent SM

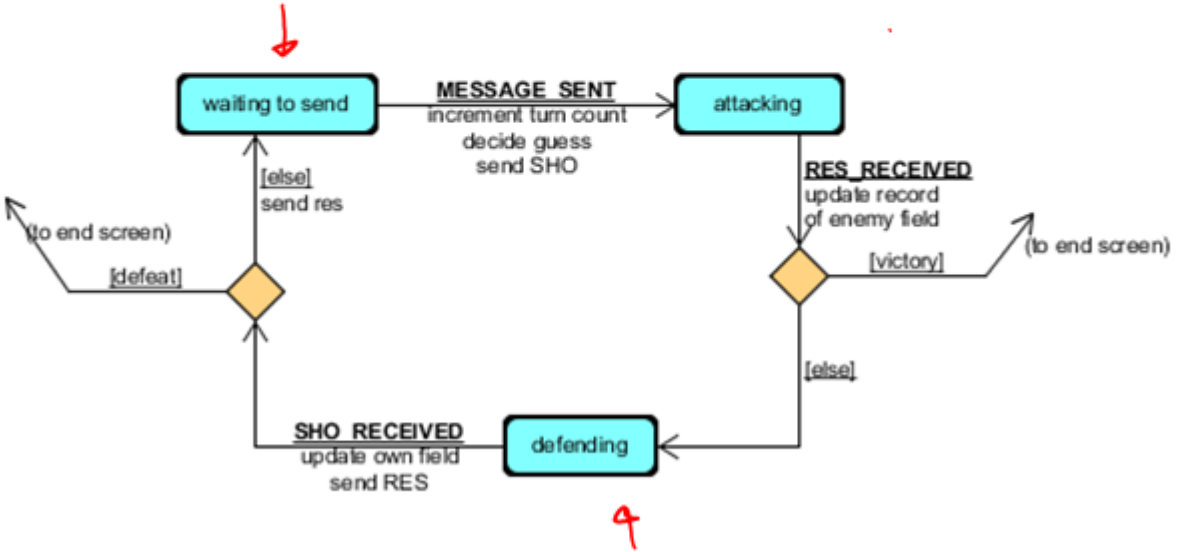

turn order negotiation
NEG
(from any state)
RESET_BUTTON
reset all data
display new game message
Start
START_BUTTON
generate A, #a
send CHA
initialize fields
CHA_RECEIVED
generate B
send ACC
initialize fields
challenging
accepting
ACC_RECEIVED
send REV
REV_RECEIVED
[heads]
(to waiting-to-send)
[tails]
(to defending)
[tails]
send SHO
(to attacking)
[cheating detected]
(to end screen)
[heads]
(to defending)
end screen
END
(from any state)
ERROR
display appropriate message to user
End screen
gameplay
PLAY
waiting to send
MESSAGE_SENT
increment turn count
decide guess
send SHO
attacking
[else]
send res
(to end screen)
[defeat]
RES_RECEIVED
update record of enemy field
[victory]
(to end screen)
[else]
SHO_RECEIVED
update own field
send RES
defending


turn order negotiation
(from any state)
RESET_BUTTON
reset all data
display new game message
Start
START_BUTTON
generate A, #a
send CHA
initialize fields
CHA_RECEIVED
generate B
send ACC
initialize fields
challenging
accepting
ACC_RECEIVED
send REV
REV_RECEIVED
[heads]
(to waiting-to-send)
[tails]
(to defending)
[tails]
send SHO
(to attacking)
[cheating detected]
(to end screen)
[heads]
(to defending)
end screen
(from any state)
ERROR
display appropriate message to user
End screen

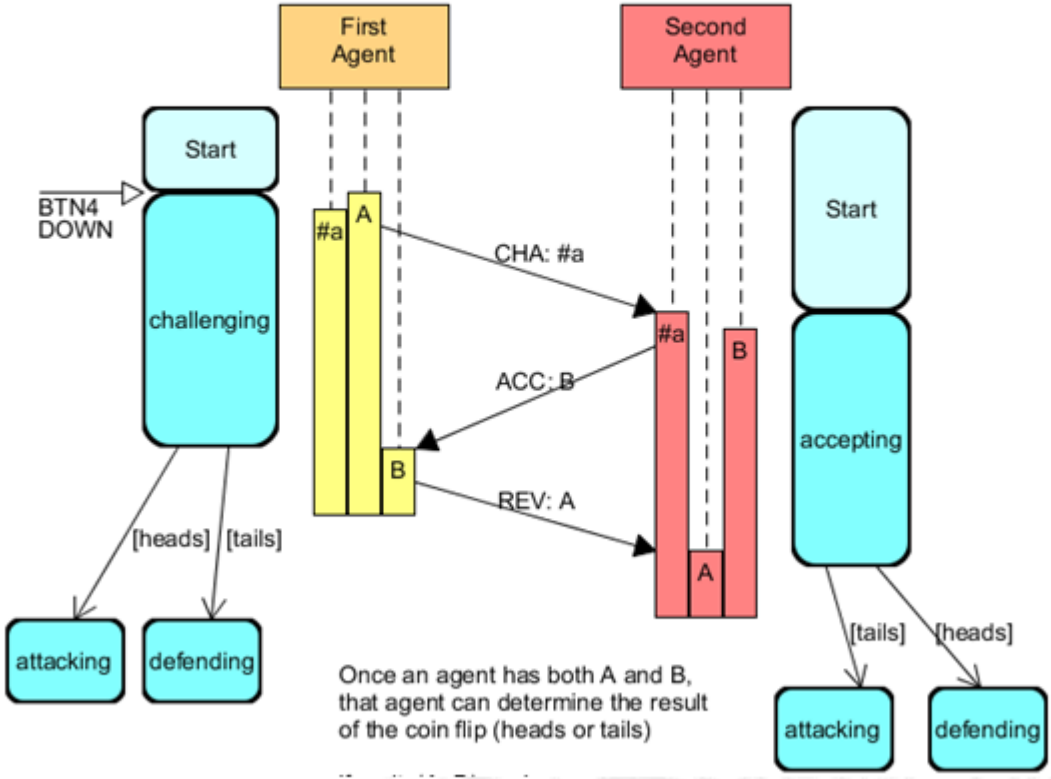

ACC_RECEIVED
send REV
[heads]
(to
waiting-to-send)
[tails]
(to defending)


REV_RECEIVED
[tails]
send SHO
(to attacking)
[cheating
detected]
(to end
screen)
[heads]
(to defending)

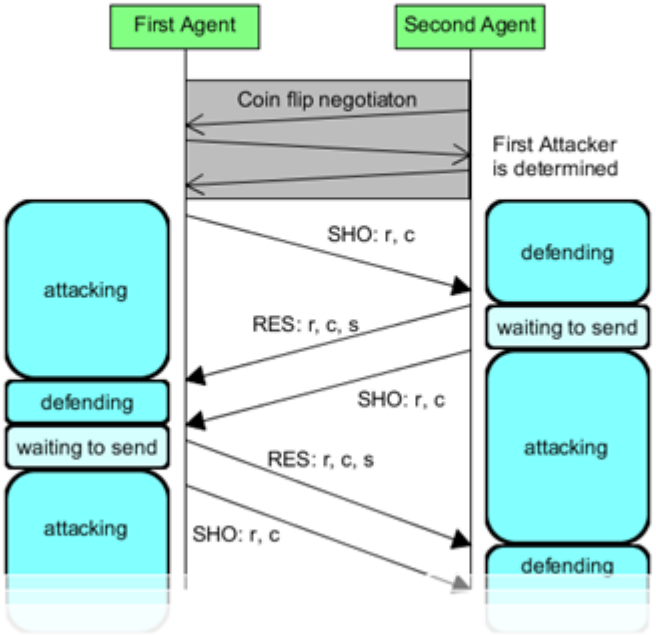
gameplay


waiting to send
MESSAGE_SENT
increment turn count
decide guess
send SHO
attacking
[else]
send res
(to end screen)
[defeat]
RES_RECEIVED
update record
of enemy field
[victory]
(to end screen)
[else]
SHO_RECEIVED
update own field
send RES
defending

Protocol sequence diagram for turn order negotiation phase:



Protocol sequence diagram for gameplay phase:



Battleboats Tips:

- Stay calm!
  - Much easier if you frontload!
- Zoom in, zoom out
- TEST TEST TEST
  - Test early, test often
- Trust!

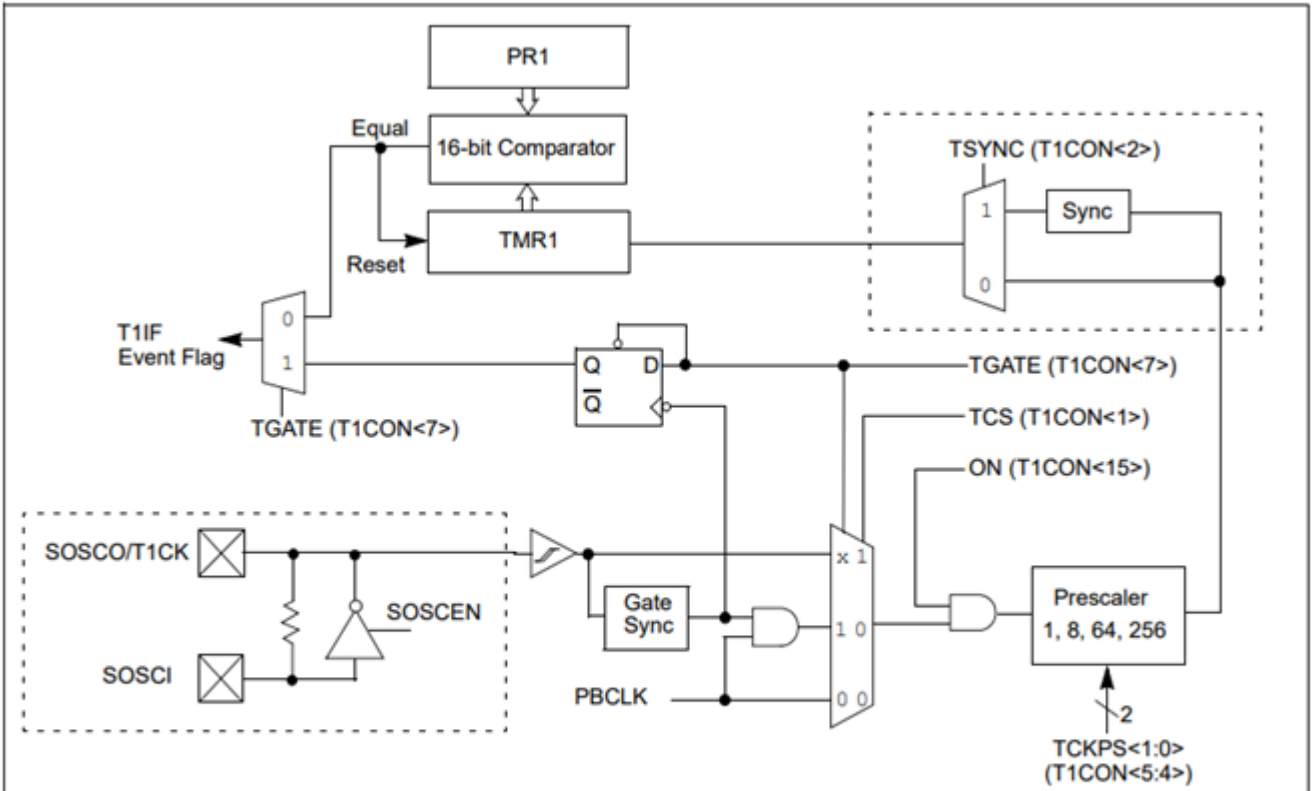
BREAK

Battleboats Extra Credit

- Implement a human agent
- Implement a cheating agent
- Implement a Very Good* AI
- Implement error recovery

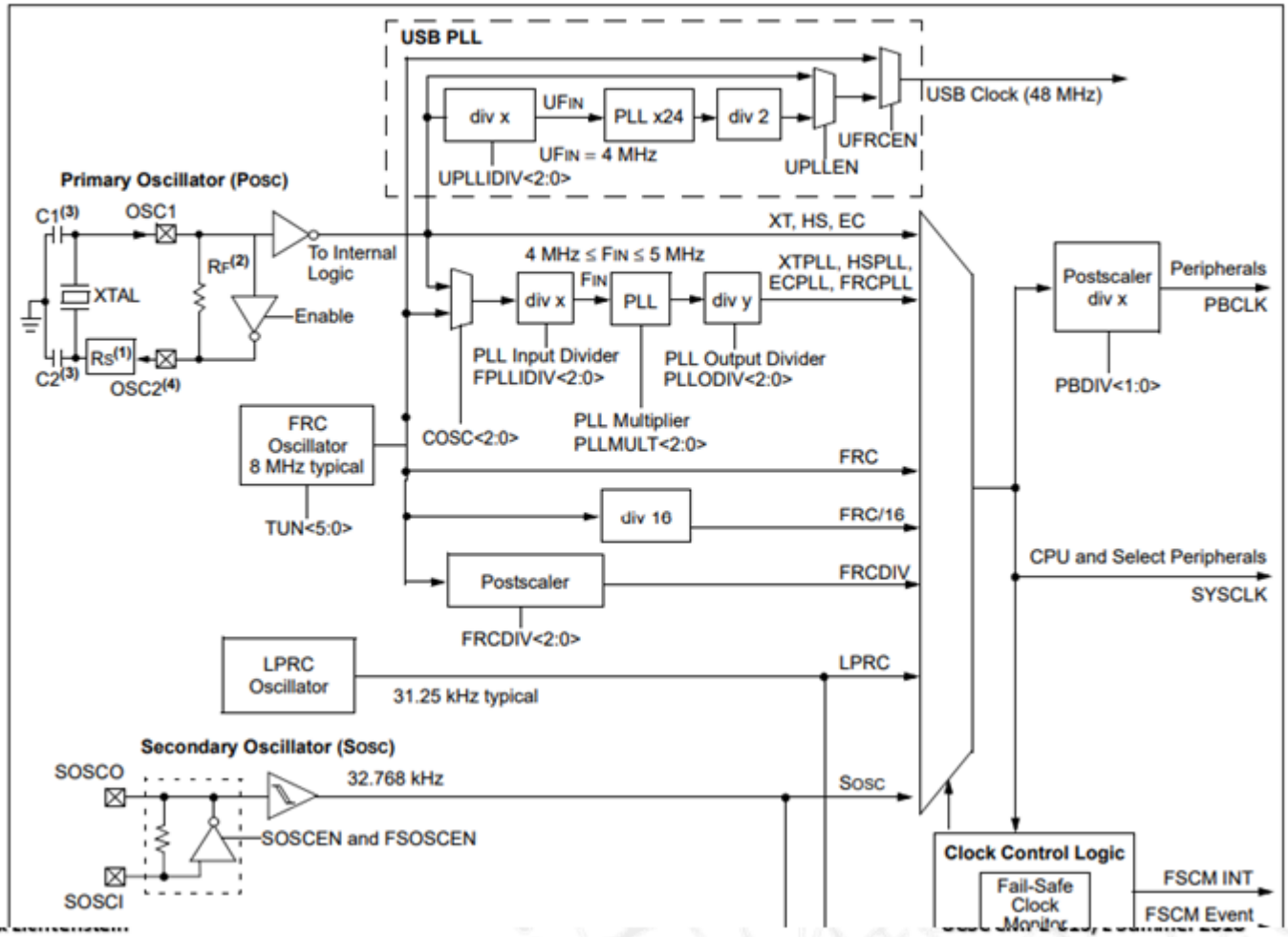
How fast can an ISR go?

FIGURE 13-1: TIMER1 BLOCK DIAGRAM(1)



Note 1: The default state of the SOSCEN (OSCCON<1>) during a device Reset is controlled by the FSOSCEN bit in Configuration Word DEVCFG1.

FIGURE 8-1: PIC32MX3XX/4XX FAMILY CLOCK DIAGRAM



bit 18-16 **PLLMULT<2:0>:** Phase-Locked Loop (PLL) Multiplier bits

The POR default is set by the FPLLMUL<2:0> bits (DEVCFG2<6:4>). Do not change these bits if the PLL is enabled. Refer to the **"Special Features"** chapter in the specific device data sheet for details.

111 = Clock is multiplied by 24
110 = Clock is multiplied by 21
101 = Clock is multiplied by 20
100 = Clock is multiplied by 19
011 = Clock is multiplied by 18
010 = Clock is multiplied by 17
001 = Clock is multiplied by 16
000 = Clock is multiplied by 15

High-Performance 32-bit RISC CPU:

- MIPS32® M4K® 32-bit core with 5-stage pipeline
- 80 MHz maximum frequency
- 1.56 DMIPS/MHz (Dhrystone 2.1) performance at 0 wait state Flash access
- Single-cycle multiply and high-performance divide unit
- MIPS16e® mode for up to 40% smaller code size
- Two sets of 32 core register files (32-bit) to reduce interrupt latency
- Prefetch Cache module to speed execution from Flash

```
void __ISR(_TIMER_1_VECTOR, IPL4AUTO) Timer1Handler(void) {
    // Clear the interrupt flag.
    INTClearFlag(INT_T1);

    // If we've exceeded the timer trigger count, trigger a timer event.
    if (++timerData.value > SWITCH_STATES()) {
        timerData.event = true;
        timerData.value = 0;
    }
}
```

What happens when an interrupt occurs?

```
!void __ISR(_TIMER_1_VECTOR, IPL4AUTO) Timer1Handler(void) {
0x9D0026F8: RDPGPR SP, SP
0x9D0026FC: MFC0 K1, EPC
0x9D002700: MFC0 K0, SRSCtl
0x9D002704: ADDIU SP, SP, -120
0x9D002708: SW K1, 116(SP)
0x9D00270C: MFC0 K1, Status
0x9D002710: SW K0, 108(SP)
0x9D002714: SW K1, 112(SP)
0x9D002718: INS K1, ZERO, 1, 15
0x9D00271C: ORI K1, K1, 4096
0x9D002720: MTC0 K1, Status
0x9D002724: SW V1, 28(SP)
0x9D002728: SW V0, 24(SP)
0x9D00272C: LW V1, 108(SP)
0x9D002730: ANDI V1, V1, 15
0x9D002734: BNE V1, ZERO, 0x9D002780
0x9D002738: NOP
0x9D00273C: SW RA, 92(SP)
0x9D002740: SW S8, 88(SP)
0x9D002744: SW T9, 84(SP)
0x9D002748: SW T8, 80(SP)
0x9D00274C: SW T7, 76(SP)
0x9D002750: SW T6, 72(SP)
0x9D002754: SW T5, 68(SP)
0x9D002758: SW T4, 64(SP)
```

```
0x9D002774: SW A1, 36(SP)
0x9D002778: SW A0, 32(SP)
0x9D00277C: SW AT, 20(SP)
0x9D002780: NOP
0x9D002784: MFLO V0
0x9D002788: SW V0, 100(SP)
0x9D00278C: MFHI V1
0x9D002790: SW V1, 96(SP)
0x9D002794: ADDU S8, SP, ZERO
!    // Clear the interrupt flag.
!    INTClearFlag(INT_T1);
0x9D002798: ADDIU A0, ZERO, 8
0x9D00279C: JAL INTClearFlag
0x9D0027A0: NOP
```