

```

1 ; Marc Dominic Cabote
2 ; CECS 460 Spring 2018
3 ; Full UART Assembly
4 ;=====
5 ; FOR DISPLAYING 0-9 AND A-Z (ALL CAPS)
6 ;=====
7 ascii_0 EQU 0030
8 ascii_1 EQU 0031
9 ascii_2 EQU 0032
10 ascii_3 EQU 0033
11 ascii_4 EQU 0034
12 ascii_5 EQU 0035
13 ascii_6 EQU 0036
14 ascii_7 EQU 0037
15 ascii_8 EQU 0038
16 ascii_9 EQU 0039
17 ascii_A EQU 0041
18 ascii_B EQU 0042
19 ascii_C EQU 0043
20 ascii_D EQU 0044
21 ascii_E EQU 0045
22 ascii_F EQU 0046
23 ascii_G EQU 0047
24 ascii_H EQU 0048
25 ascii_I EQU 0049
26 ascii_J EQU 004A
27 ascii_K EQU 004B
28 ascii_L EQU 004C
29 ascii_M EQU 004D
30 ascii_N EQU 004E
31 ascii_O EQU 004F
32 ascii_P EQU 0050
33 ascii_Q EQU 0051
34 ascii_R EQU 0052
35 ascii_S EQU 0053
36 ascii_T EQU 0054
37 ascii_U EQU 0055
38 ascii_V EQU 0056
39 ascii_W EQU 0057
40 ascii_X EQU 0058
41 ascii_Y EQU 0059
42 ascii_Z EQU 005A
43 ;=====
44 ; FOR DISPLAYING SYMBOLS and LINE MANIPULATION
45 ;=====
46 ascii_BCKSPC EQU 0008 ; <-
47 ascii_TAB EQU 0009 ; TAB
48 ascii_LF EQU 000A ; Line Feed
49 ascii_CR EQU 000D ; Carriage Return
50 ascii_SPC EQU 0020 ; SPACE
51 ascii_asterisk EQU 002A ; *
52 ascii_DASH EQU 002D ; "-"
53 ascii_DOT EQU 002E ; .
54 ascii_AT EQU 0040 ; @
55
56 ;=====
57 ; CONSTANTS
58 ;=====
59 ZERO EQU 0000
60 ONE EQU 0001
61
62 ;=====
63 ; PORTS
64 ;=====
65 DATA_PORT EQU 0000
66 STATUS_PORT EQU 0001
67
68 ;=====
69 ; REGISTERS

```

```

70 ;=====
71 CHAR_REG EQU R1
72 CHAR_INDEX EQU R2
73 CHAR_COUNT EQU R3 ;for reading through registers
74 LED_COUNT EQU R4
75 DELAY_COUNT EQU R5
76 LEDS EQU R6
77 TEMP EQU R7
78 RB EQU RB
79 RD EQU RD
80 RE EQU RE
81 STATUS EQU R8
82 COUNT1 EQU R9 ;for displaying dashes
83 COUNT_CHAR EQU RA ;for counting characters
84 OUT_FLAG EQU RC ;print flag
85 DATA_IN EQU R0 ;data received
86
87
88
89
90 ;=====
91 ; INITIALIZATION ; banner, prompt, hometown
92 ;=====
93 START
94 ;=====
95 ; CHAR INIT
96 ;=====
97 LOAD CHAR_REG, ZERO
98 LOAD CHAR_COUNT, ZERO
99 LOAD COUNT1, ZERO
100 LOAD CHAR_INDEX, ZERO
101 LOAD TEMP, ZERO
102 LOAD RD, ZERO
103 LOAD RE, ZERO
104 LOAD STATUS, ZERO
105 LOAD OUT_FLAG, ONE ;initialized to output the banner at first
106 LOAD COUNT_CHAR, ZERO
107 ;=====
108 ; LED INIT
109 ;=====
110 LOAD LEDS, ONE
111 LOAD DELAY_COUNT, ZERO
112
113 ;=====
114 ; BANNER INIT;
115 ; -----
116 ; . MARC CABOTE .
117 ; -----
118 ;=====
119
120 LOAD CHAR_REG, ascii_DASH
121 DASHBEGIN
122 STORE CHAR_REG, CHAR_INDEX
123 ADD CHAR_INDEX, ONE
124 ADD COUNT1, ONE
125 COMP COUNT1, 0014 ; output 20 dashes
126 JUMPC DASHBEGIN
127
128 LOAD COUNT1, ZERO
129
130 LOAD CHAR_REG, ascii_CR
131 STORE CHAR_REG, 0015
132
133 LOAD CHAR_REG, ascii_LF
134 STORE CHAR_REG, 0016
135
136 LOAD CHAR_REG, ascii_DOT
137 STORE CHAR_REG, 0017
138

```

139		LOAD	CHAR_REG,	ascii_SPC	
140		STORE	CHAR_REG,	0018	
141					
142		LOAD	CHAR_REG,	ascii_SPC	
143		STORE	CHAR_REG,	0019	
144					
145		LOAD	CHAR_REG,	ascii_SPC	
146		STORE	CHAR_REG,	001A	
147					
148		LOAD	CHAR_REG,	ascii_M	
149		STORE	CHAR_REG,	001B	
150					
151		LOAD	CHAR_REG,	ascii_A	
152		STORE	CHAR_REG,	001C	
153					
154		LOAD	CHAR_REG,	ascii_R	
155		STORE	CHAR_REG,	001D	
156					
157		LOAD	CHAR_REG,	ascii_C	
158		STORE	CHAR_REG,	001E	
159					
160		LOAD	CHAR_REG,	ascii_SPC	
161		STORE	CHAR_REG,	001F	
162					
163		LOAD	CHAR_REG,	ascii_C	
164		STORE	CHAR_REG,	0020	
165					
166		LOAD	CHAR_REG,	ascii_A	
167		STORE	CHAR_REG,	0021	
168					
169		LOAD	CHAR_REG,	ascii_B	
170		STORE	CHAR_REG,	0022	
171					
172		LOAD	CHAR_REG,	ascii_O	
173		STORE	CHAR_REG,	0023	
174					
175		LOAD	CHAR_REG,	ascii_T	
176		STORE	CHAR_REG,	0024	
177					
178		LOAD	CHAR_REG,	ascii_E	
179		STORE	CHAR_REG,	0025	
180					
181		LOAD	CHAR_REG,	ascii_SPC	
182		STORE	CHAR_REG,	0026	
183					
184		LOAD	CHAR_REG,	ascii_SPC	
185		STORE	CHAR_REG,	0027	
186					
187		LOAD	CHAR_REG,	ascii_SPC	
188		STORE	CHAR_REG,	0028	
189					
190		LOAD	CHAR_REG,	ascii_SPC	
191		STORE	CHAR_REG,	0029	
192					
193		LOAD	CHAR_REG,	ascii_DOT	
194		STORE	CHAR_REG,	002A	
195					
196		LOAD	CHAR_REG,	ascii_CR	
197		STORE	CHAR_REG,	002B	
198					
199		LOAD	CHAR_REG,	ascii_LF	
200		STORE	CHAR_REG,	002C	
201					
202		LOAD	CHAR_REG,	ascii_DASH	
203		LOAD	CHAR_INDEX,	002D	;to start dash at 2D
204	DASHEND				
205		STORE	CHAR_REG,	CHAR_INDEX	
206		ADD	CHAR_INDEX,	ONE	
207		ADD	COUNT1,	ONE	

```

208      COMP      COUNT1,      0014 ; output 20 dashes til 0x40
209      JUMPC     DASHEND
210
211      ;=====
212      ;      NEW LINE
213      ;=====
214
215
216      LOAD      CHAR_REG,      ascii_CR
217      STORE     CHAR_REG,      0041
218
219      LOAD      CHAR_REG,      ascii_LF
220      STORE     CHAR_REG,      0042
221
222      LOAD      COUNT1,      ZERO
223
224      ;=====
225      ;      HOMETOWN
226      ;=====
227
228      LOAD      CHAR_REG,      ascii_H
229      STORE     CHAR_REG,      0043
230
231      LOAD      CHAR_REG,      ascii_O
232      STORE     CHAR_REG,      0044
233
234      LOAD      CHAR_REG,      ascii_M
235      STORE     CHAR_REG,      0045
236
237      LOAD      CHAR_REG,      ascii_E
238      STORE     CHAR_REG,      0046
239
240      LOAD      CHAR_REG,      ascii_T
241      STORE     CHAR_REG,      0047
242
243      LOAD      CHAR_REG,      ascii_O
244      STORE     CHAR_REG,      0048
245
246      LOAD      CHAR_REG,      ascii_W
247      STORE     CHAR_REG,      0049
248
249      LOAD      CHAR_REG,      ascii_N
250      STORE     CHAR_REG,      004A
251
252      LOAD      CHAR_REG,      ascii_SPC
253      STORE     CHAR_REG,      004B
254
255      LOAD      CHAR_REG,      ascii_DASH
256      STORE     CHAR_REG,      004C
257
258      LOAD      CHAR_REG,      ascii_SPC
259      STORE     CHAR_REG,      004D
260
261      LOAD      CHAR_REG,      ascii_L
262      STORE     CHAR_REG,      004E
263
264      LOAD      CHAR_REG,      ascii_O
265      STORE     CHAR_REG,      004F
266
267      LOAD      CHAR_REG,      ascii_S
268      STORE     CHAR_REG,      0050
269
270      LOAD      CHAR_REG,      ascii_A
271      STORE     CHAR_REG,      0051
272
273      LOAD      CHAR_REG,      ascii_N
274      STORE     CHAR_REG,      0052
275
276      LOAD      CHAR_REG,      ascii_G

```

```

277         STORE    CHAR_REG,    0053
278
279         LOAD     CHAR_REG,    ascii_E
280         STORE    CHAR_REG,    0054
281
282         LOAD     CHAR_REG,    ascii_L
283         STORE    CHAR_REG,    0055
284
285         LOAD     CHAR_REG,    ascii_E
286         STORE    CHAR_REG,    0056
287
288         LOAD     CHAR_REG,    ascii_S
289         STORE    CHAR_REG,    0057
290
291         LOAD     CHAR_REG,    ascii_CR
292         STORE    CHAR_REG,    0058
293
294         LOAD     CHAR_REG,    ascii_LF
295         STORE    CHAR_REG,    0059
296
297         AND      CHAR_INDEX, ZERO    ;reset char index
298
299         ;=====
300         ;    BACKSPACE
301         ;=====
302
303         LOAD     CHAR_REG,    ascii_BCKSPC
304         STORE    CHAR_REG,    005A
305
306         LOAD     CHAR_REG,    ascii_SPC
307         STORE    CHAR_REG,    005B
308
309         LOAD     CHAR_REG,    ascii_BCKSPC
310         STORE    CHAR_REG,    005C
311
312         ;=====
313         ;    PROMPT
314         ;=====
315         LOAD     CHAR_REG,    ascii_I
316         STORE    CHAR_REG,    00B0
317
318         LOAD     CHAR_REG,    ascii_N
319         STORE    CHAR_REG,    00B1
320
321         LOAD     CHAR_REG,    ascii_P
322         STORE    CHAR_REG,    00B2
323
324         LOAD     CHAR_REG,    ascii_U
325         STORE    CHAR_REG,    00B3
326
327         LOAD     CHAR_REG,    ascii_T
328         STORE    CHAR_REG,    00B4
329
330         LOAD     CHAR_REG,    ascii_DASH
331         STORE    CHAR_REG,    00B5
332
333         LOAD     CHAR_REG,    ascii_DASH
334         STORE    CHAR_REG,    00B6
335
336         ENINT
337         ;=====
338         ;                OUTPUT a walking LED
339         ;=====
340 MAIN
341         ADD      LED_COUNT,    ONE
342         ADDC     DELAY_COUNT,    ZERO
343         COMP     DELAY_COUNT,    0007 ; delay
344         JUMPC    DONE_LED
345

```

```

346         LOAD     LED_COUNT,      ZERO
347         LOAD     DELAY_COUNT,     ZERO
348         RL       LEDS ;rotate LEDS
349
350 DONE_LED
351         OUTPUT    LEDS,             0002
352         JUMP      MAIN
353 ;=====
354 ; Interrupt Service Routine
355 ; Starts at address 0300
356 ; CHECK RXRDY
357 ; CHECK TXRDY
358 ;=====
359
360         ADDRESS   0300
361 ISR
362         ;FETCH    CHAR_REG,         COUNT
363         ;OUTPUT   CHAR_REG,        DATA_PORT
364         ;ADD      COUNT,           ONE
365
366         INPUT     STATUS,          STATUS_PORT
367         AND       STATUS,          0003
368
369         COMP      STATUS,          0003      ;IF BOTH RXRDY and TXRDY
370         JUMPZ     BOTH_RDY
371
372         COMP      STATUS,          0002      ;IF TXRDY
373         CALLZ     TX_F
374
375         COMP      STATUS,          0001      ;IF RXRDY
376         CALLZ     RX_F
377
378         RETEN
379
380 BOTH_RDY
381         CALL      TX_F
382         CALL      RX_F
383         RETEN
384
385
386 ;=====
387 ; ADDRESS FOR BIN_TO_ASCII
388 ;=====
389 BIN_TO_ASCII
390         LOAD      RE, COUNT_CHAR
391
392         LOAD      RD, 000A      ;RD<-10
393         CALL      FIND_IT
394         ADD       RB, 0030      ;convert to ascii_hex
395         STORE     RB, 00A0      ;store counter at 00A0
396
397         ADD       RE, 0030      ;convert to ascii_hex
398         STORE     RE, 00A1      ;store counter at 00A1
399
400         RETURN
401 ;=====
402 ; BIN_TO_ASCII FIND_IT FUNCTION
403 ;=====
404 FIND_IT
405         LOAD      RB, 0000      ;RB<-0000
406 NOT_DONE
407         SUB       RE, RD      ; RE<-RE-RD
408         JUMPC     RESTORE      ; if there is a carry restore RE
409         ADD       RB, 0001      ; increment RB
410         JUMP      NOT_DONE      ; keep subtracting
411 RESTORE
412         ADD       RE, RD      ; restore last value
413         RETURN      ; return bin to ascii routine
414 ;=====

```

```

415 ; TX
416 ;=====
417 TX_F
418     COMP     OUT_FLAG,    ZERO
419     RETURNZ
420
421     FETCH    TEMP,        CHAR_COUNT
422     OUTPUT    TEMP,        DATA_PORT
423     ADD      CHAR_COUNT, ONE
424
425     COMP     OUT_FLAG,    ONE
426     JUMPZ    BANNER_OUT
427
428     COMP     OUT_FLAG,    0002
429     JUMPZ    PROMPT_OUT
430
431     COMP     OUT_FLAG,    0003
432     JUMPZ    HOMETOWN_OUT
433
434     COMP     OUT_FLAG,    0004
435     JUMPZ    COUNT_OUT
436
437     COMP     OUT_FLAG,    0005
438     JUMPZ    BACKSPACE_OUT
439
440     COMP     OUT_FLAG,    0006
441     JUMPZ    NEW_LINE
442
443     RETURN
444
445 BANNER_OUT
446     COMP     CHAR_COUNT, 0043 ;address where banner ends
447     RETURNZ
448     LOAD     OUT_FLAG,    0002 ;prompt
449     LOAD     CHAR_COUNT, 00B0 ;where prompt starts
450     RETURN
451
452 PROMPT_OUT
453     COMP     CHAR_COUNT, 00B7 ;address where prompt ends
454     RETURNZ
455     LOAD     OUT_FLAG,    0000 ;wait for RX
456     RETURN
457
458 HOMETOWN_OUT
459     COMP     CHAR_COUNT, 005A ;address where hometown ends
460     RETURNZ
461     LOAD     CHAR_COUNT, 0041 ;address where prompt starts
462     LOAD     OUT_FLAG,    0002 ;prompt
463     LOAD     CHAR_COUNT, 00B0 ;where prompt starts
464     RETURN
465
466 COUNT_OUT
467     COMP     CHAR_COUNT, 00A2 ;address where character count ends
468     RETURNZ
469     LOAD     CHAR_COUNT, 0041 ;new line
470     LOAD     OUT_FLAG,    0006
471     RETURN
472
473 BACKSPACE_OUT
474     COMP     CHAR_COUNT, 005D ;address where backspace ends
475     RETURNZ
476     LOAD     OUT_FLAG,    0000 ;wait for RX
477     RETURN
478
479 NEW_LINE
480     COMP     CHAR_COUNT, 0043 ;address where new line ends
481     RETURNZ
482     LOAD     CHAR_COUNT, 0041 ;address where prompt starts
483     LOAD     OUT_FLAG,    0002 ;prompt

```

```

484         LOAD      CHAR_COUNT, 00B0      ;where prompt starts
485         RETURN
486
487 ;=====
488 ;                               RX
489 ;=====
490 RX_F
491         COMP      OUT_FLAG,  ZERO      ;wait for input
492         RETURNNZ
493
494         INPUT     DATA_IN,  ZERO
495         COMP      DATA_IN,  ZERO      ;if no input
496         RETURNZ
497
498         COMP      DATA_IN,  ascii_ASTERISK ; if *
499         JUMPZ     GO_HOMETOWN
500
501         COMP      DATA_IN,  ascii_AT      ; if @
502         JUMPZ     GO_CHARCOUNT
503
504         COMP      DATA_IN,  ascii_BCKSPC  ; if backspace
505         JUMPZ     GO_BACKSPACE
506
507         COMP      DATA_IN,  ascii_CR      ; if return
508         JUMPZ     GO_NEWLINE
509
510         ADD       COUNT_CHAR, ONE          ;increment char counter
511         OUTPUT    DATA_IN,  DATA_PORT    ;output char counter
512         COMP      COUNT_CHAR, 0028        ;conter cannot exceed 40
513         JUMPZ     GO_NEWLINE
514         RETURN
515
516 GO_HOMETOWN
517         LOAD      OUT_FLAG,  0003      ;hometown out
518         LOAD      CHAR_COUNT, 0043      ;address where hometown starts
519         LOAD      TEMP,  ZERO
520         OUTPUT    TEMP,  DATA_PORT
521         LOAD      COUNT_CHAR, ZERO      ;characters counter
522         RETURN
523
524 GO_CHARCOUNT
525         CALL      BIN_TO_ASCII
526         LOAD      OUT_FLAG,  0004      ;char count out
527         LOAD      CHAR_COUNT, 00A0      ;address where counter starts
528         LOAD      TEMP,  ZERO
529         OUTPUT    TEMP,  DATA_PORT
530         LOAD      COUNT_CHAR, ZERO      ;characters counter
531         RETURN
532
533 GO_BACKSPACE
534         COMP      COUNT_CHAR, ZERO
535         RETURNZ
536         LOAD      OUT_FLAG,  0005      ;backspace out
537         LOAD      CHAR_COUNT, 005A      ;address where backspace starts
538         LOAD      TEMP,  ZERO
539         OUTPUT    TEMP,  DATA_PORT
540         SUB       COUNT_CHAR, 0001      ;decrement characters counter
541         RETURN
542
543 GO_NEWLINE
544         LOAD      OUT_FLAG,  0006      ;newline out
545         LOAD      CHAR_COUNT, 0041      ;address where newline begins
546         LOAD      TEMP,  ZERO
547         OUTPUT    TEMP,  DATA_PORT
548         LOAD      COUNT_CHAR, ZERO
549         RETURN
550
551 ;=====
552 ;   ISR VECTORED THROUGH 0FFE

```



```
553 ;=====
554 ADDRESS OFFE
555 ENDIT
556 JUMP ISR
557 END
558 ;=====
559 ;
560 ;=====
```