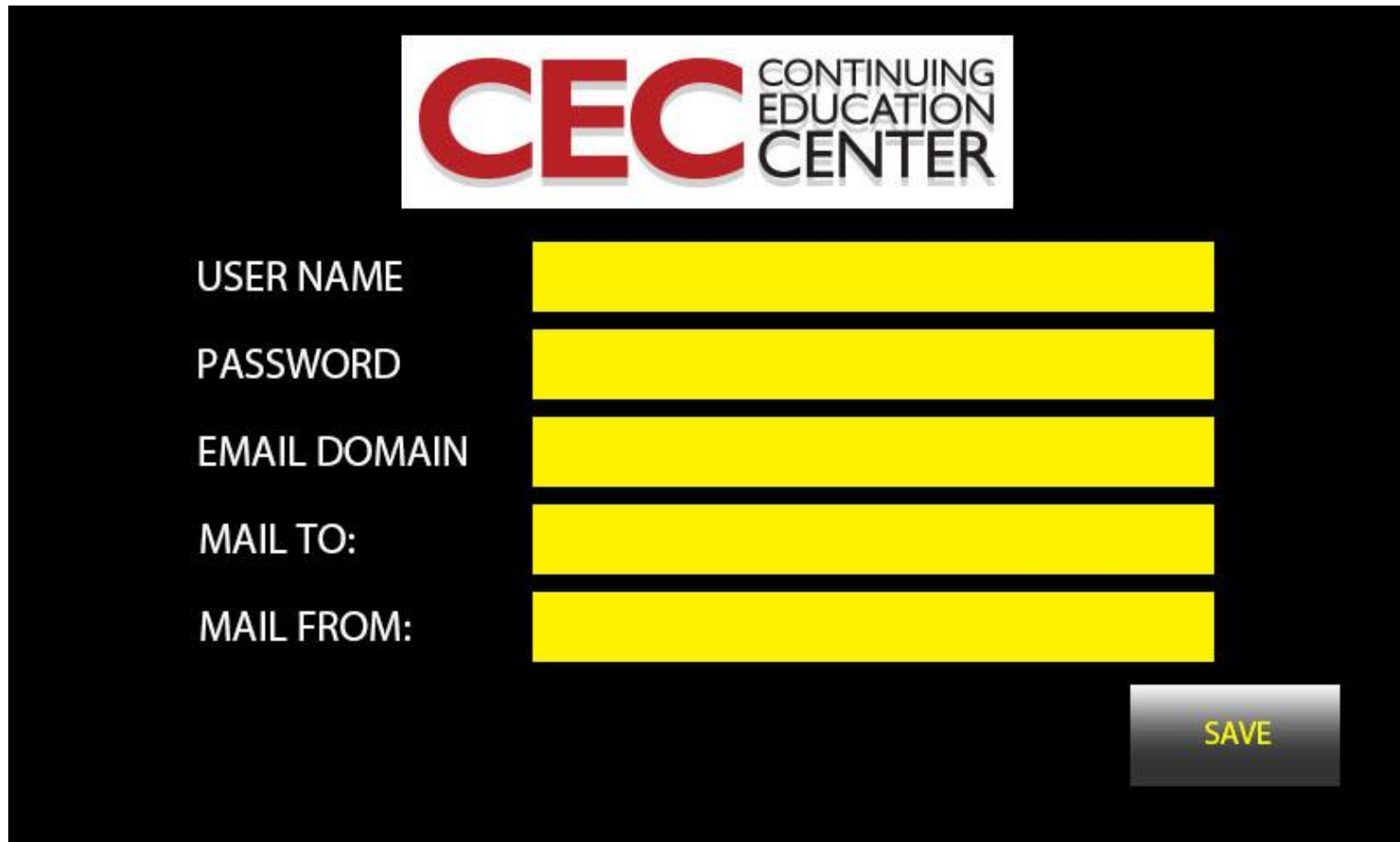


Essential Coding Techniques for Hardware Engineers



CEC CONTINUING
EDUCATION
CENTER

USER NAME

PASSWORD

EMAIL DOMAIN

MAIL TO:

MAIL FROM:

SAVE

Coding the Touch Display Interface

January 29, 2019

Fred Eady

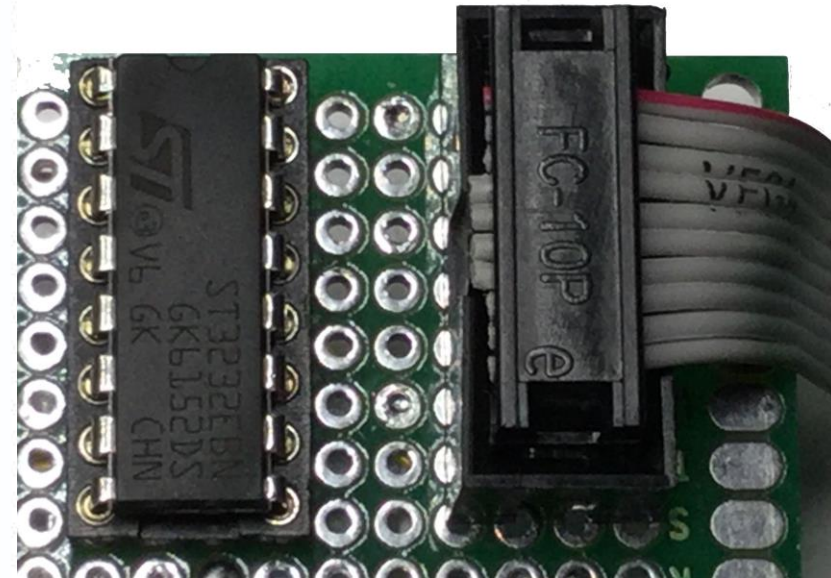
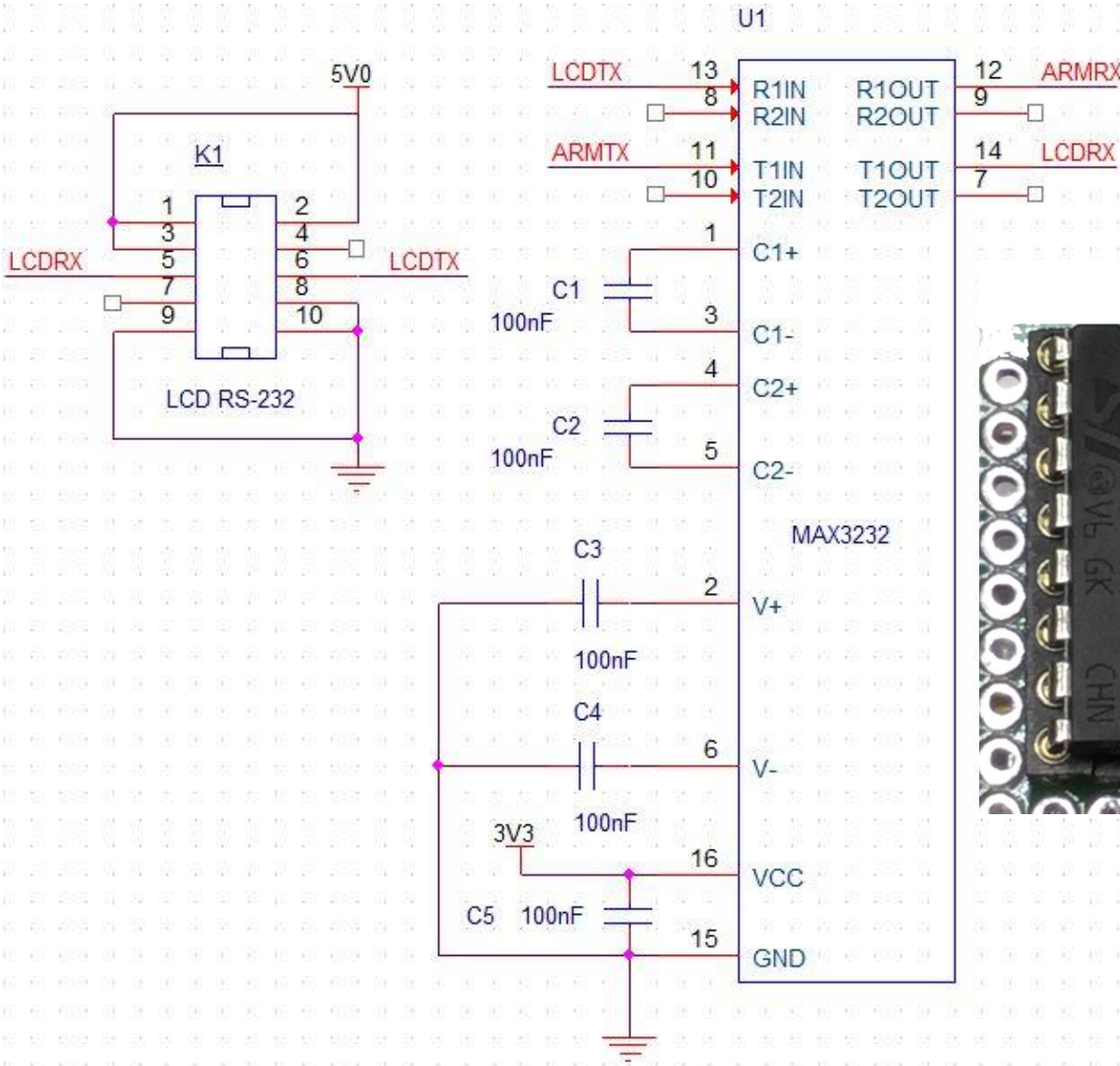
AGENDA

- Touch Display – The Physical Interface
- Touch Display – The Logical Interface
- Day 2 Summary



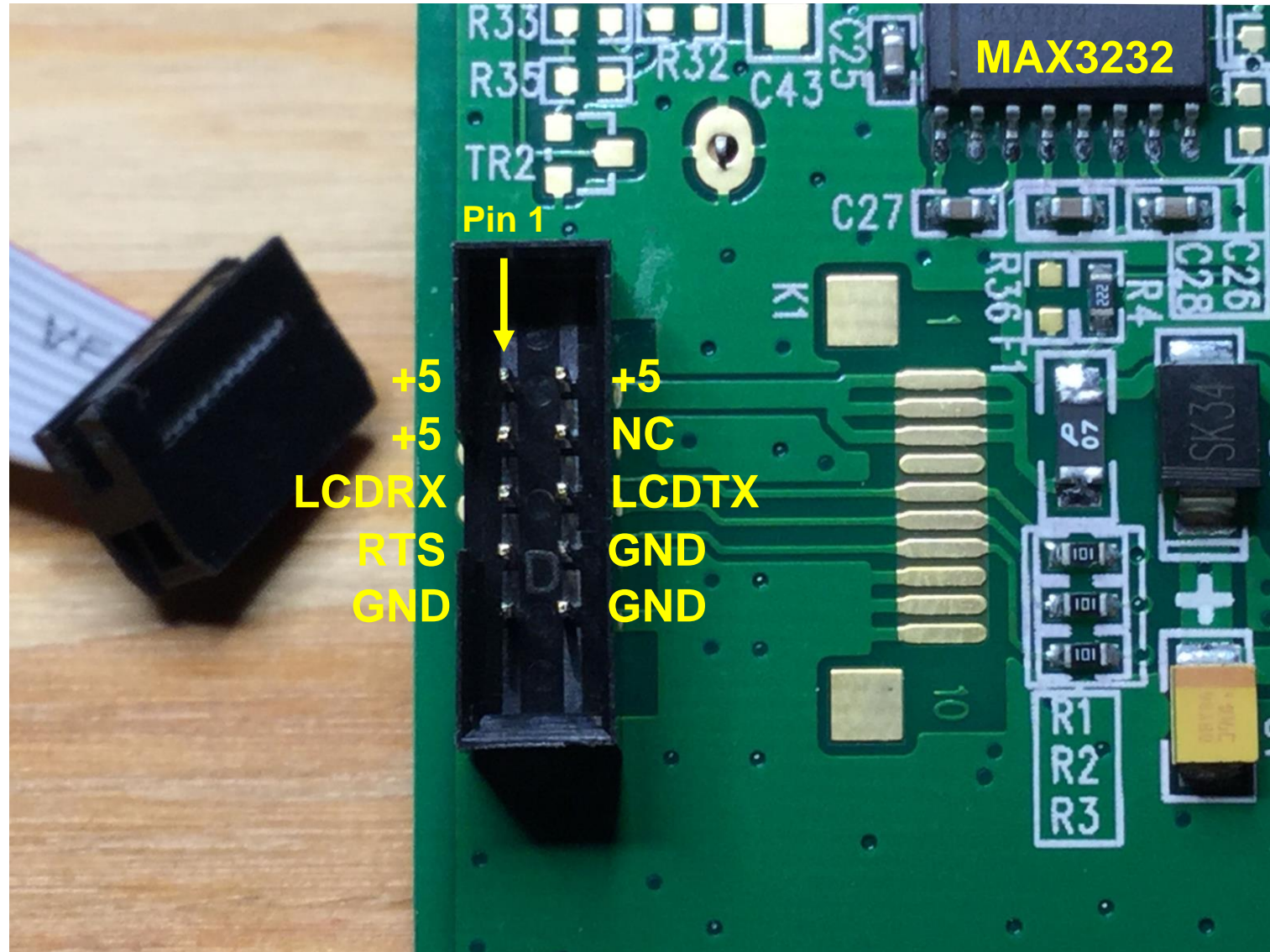
Essential Coding Techniques for Hardware Engineers

Touch Display - The Physical Interface: ARM



Essential Coding Techniques for Hardware Engineers

Touch Display - The Physical Interface: Touch Display



Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: UART1 Init

```
80 //*****
81 /** TYPEDEFS
82 //*****
83 SPI_HandleTypeDef hspil;
84
85 UART_HandleTypeDef huart1;
86 UART_HandleTypeDef huart2;
```

```
869 static void MX_USART1_UART_Init(void)
870 {
871     huart1.Instance = USART1;
872     huart1.Init.BaudRate = 115200;
873     huart1.Init.WordLength = UART_WORDLENGTH_8B;
874     huart1.Init.StopBits = UART_STOPBITS_1;
875     huart1.Init.Parity = UART_PARITY_NONE;
876     huart1.Init.Mode = UART_MODE_TX_RX;
877     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
878     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
879     huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
880     huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
881     huart1.Mask = 0x00FF;
882     if (HAL_UART_Init(&huart1) != HAL_OK)
883     {
884         Error_Handler();
885     }
886 }
887
```

```
727
728     __HAL_UART_ENABLE_IT(&huart1, UART_IT_RXNE);
729
```

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Display Data/Address Definitions

```
88 typedef struct{
89     uint8_t    uname[128];
90     uint8_t    pword[128];
91     uint8_t    domain[128];
92     uint8_t    emto[128];
93     uint8_t    emfrm[128];
94 }EMAILDATA;
95 EMAILDATA emaildata;
96
97 typedef struct{
98     uint8_t    cmd;
99     uint8_t    pageIDh;
100    uint8_t    pageIDl;
101    uint8_t    keyID;
102    uint8_t    vpAddr3;
103    uint8_t    vpAddr2;
104    uint8_t    vpAddr1;
105    uint8_t    vpAddr0;
106    uint32_t    vpAddr;
107    //uint8_t    vpN16h;
108    //uint8_t    vpN16l;
109    //uint16_t    vpN16;
110    //uint8_t    vpNascii[5];
111    uint8_t    vpStr[128];
112 }TOUCHRESPONSE;
113 TOUCHRESPONSE resp;
```

```
114 //*****
115 /* DISPLAY PAGE DEFINITIONS
116 //*****
117 #define displaypage0000 0x0000
118 //*****
119 /* DISPLAY TOUCH KEY ID DEFINITIONS
120 //*****
121 #define tksave_page0000 0x05
122 //*****
123 /* DISPLAY STRING MEMORY ADDRESSES
124 //*****
125 #define username      0x00000080
126 #define passwrld     0x00000100
127 #define edomain      0x00000180
128 #define mailto       0x00000200
129 #define mailfrom     0x00000280
```


Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Ring Buffer

```
176 //*****
177 /** USART RECEIVE BUFFERS SETUP
178 /** 1,2,4,8,16,32,64,128 or 256 uint8_tS
179 //*****
180 #define USART1_RX_BUFFER_SIZE 256
181 #define USART1_RX_BUFFER_MASK ( USART1_RX_BUFFER_SIZE - 1 )
182 uint8_t USART1_RxBuf[USART1_RX_BUFFER_SIZE];
183 uint8_t USART1_RxTail;
184 uint8_t USART1_RxHead;
```

The screenshot shows the Hercules SETUP utility interface. The 'Serial' tab is active, displaying 'Received/Sent data' with the following log: 'Serial port COM23 opened', '1123', 'Serial port COM23 closed', 'Serial port COM23 opened', and 'T22222H'. The 'Serial' configuration shows 'Name: COM23' and 'Baud: 115200'. A 'Watch 1' window displays the following data:

Name	Value	Type
USART1_RxBuf	0x20000494 USART1_R...	unsigned char[256]
USART1_RxTail	0x00	unsigned char
USART1_RxHead	0x07	unsigned char

The memory dump on the right shows the contents of the USART1_RxBuf array. The 'Tail' pointer is at index 0 (0x00) and the 'Head' pointer is at index 7 (0x48 'H').

Index	Value	Type
[0]	0x00	unsigned char
[1]	0x54 'T'	unsigned char
[2]	0x32 '2'	unsigned char
[3]	0x32 '2'	unsigned char
[4]	0x32 '2'	unsigned char
[5]	0x32 '2'	unsigned char
[6]	0x32 '2'	unsigned char
[7]	0x48 'H'	unsigned char
[8]	0x00	unsigned char
[9]	0x00	unsigned char
[10]	0x00	unsigned char
[11]	0x00	unsigned char
[12]	0x00	unsigned char
[13]	0x00	unsigned char
[14]	0x00	unsigned char
[15]	0x00	unsigned char
[16]	0x00	unsigned char
[17]	0x00	unsigned char
[18]	0x00	unsigned char
[19]	0x00	unsigned char
[20]	0x00	unsigned char
[21]	0x00	unsigned char
[22]	0x00	unsigned char
[23]	0x00	unsigned char
[24]	0x00	unsigned char

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: UART1 Interrupt Handler

```
170 void USART1_IRQHandler(void)
171 {
172     //HAL_UART_IRQHandler(&huart1);
173     uint32_t  isrflags  = READ_REG(huart1.Instance->ISR);
174     uint32_t  crlits    = READ_REG(huart1.Instance->CR1);
175     uint32_t  errorflags;
176     uint16_t  data;
177     uint8_t   tmphead;
178     uint16_t  dataMask = huart1.Mask;
179
180     errorflags = (isrflags & (uint32_t)(USART_ISR_PE | USART_ISR_FE | USART_ISR_ORE | USART_ISR_NE));
181     if (errorflags == RESET)
182     {
183         /* UART in mode Receiver -----*/
184         if(((isrflags & USART_ISR_RXNE) != RESET) && ((crlits & USART_CR1_RXNEIE) != RESET))
185         {
186             // read byte from UART
187             data = (uint16_t) READ_REG(huart1.Instance->RDR);
188
189             // calculate buffer index
190             tmphead = ( USART1_RxHead + 1 ) & USART1_RX_BUFFER_MASK;
191             // store new index
192             USART1_RxHead = tmphead;
193
194             if ( tmphead == USART1_RxTail )
195             {
196                 // ERROR! Receive buffer overflow
197             }
198             // store received data in ring buffer
199             USART1_RxBuf[tmphead] = (uint8_t)((uint8_t)dataMask & data);
200         }
201     }
202 }
203
```


Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Read the Ring

```
480 //*****
481 //*   CHECK FOR CHARACTER IN RING
482 //*****
483 uint8_t CharInRing(void)
484 {
485     return(USART1_RxHead != USART1_RxTail);
486 }
487 //*****
488 //*   GET BYTE FROM RX RING BUFFER
489 //*****
490 uint8_t readring(void)
491 {
492     __HAL_UART_DISABLE_IT(&huart1,UART_IT_RXNE);
493     uint8_t tmptail;
494     // wait for incoming data
495     while ( USART1_RxHead == USART1_RxTail );
496     // calculate buffer index
497     tmptail = ( USART1_RxTail + 1 ) & USART1_RX_BUFFER_MASK;
498     // store new index
499     USART1_RxTail = tmptail;
500     __HAL_UART_ENABLE_IT(&huart1,UART_IT_RXNE);
501     // return data
502     return USART1_RxBuf[tmptail];
503 }
```

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Structure

Sequence	Code	Code Type	Description
1	0xAA	Packet Header	1 Byte
2	CMD Code	Command Code	1 Byte
3	Param/Data	Parameter or Data	Multi-Byte
:	:	:	
:	:	:	
Nth-3	0xCC	Packet Tail	4 Bytes
Nth-2	0x33		
Nth-1	0xC3		
Nth	0x3C		

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

PG0000-displaypage0000

CEC CONTINUING EDUCATION CENTER

USER NAME User Name p

PASSWORD Password p

EMAIL DOMAIN Email Domain p

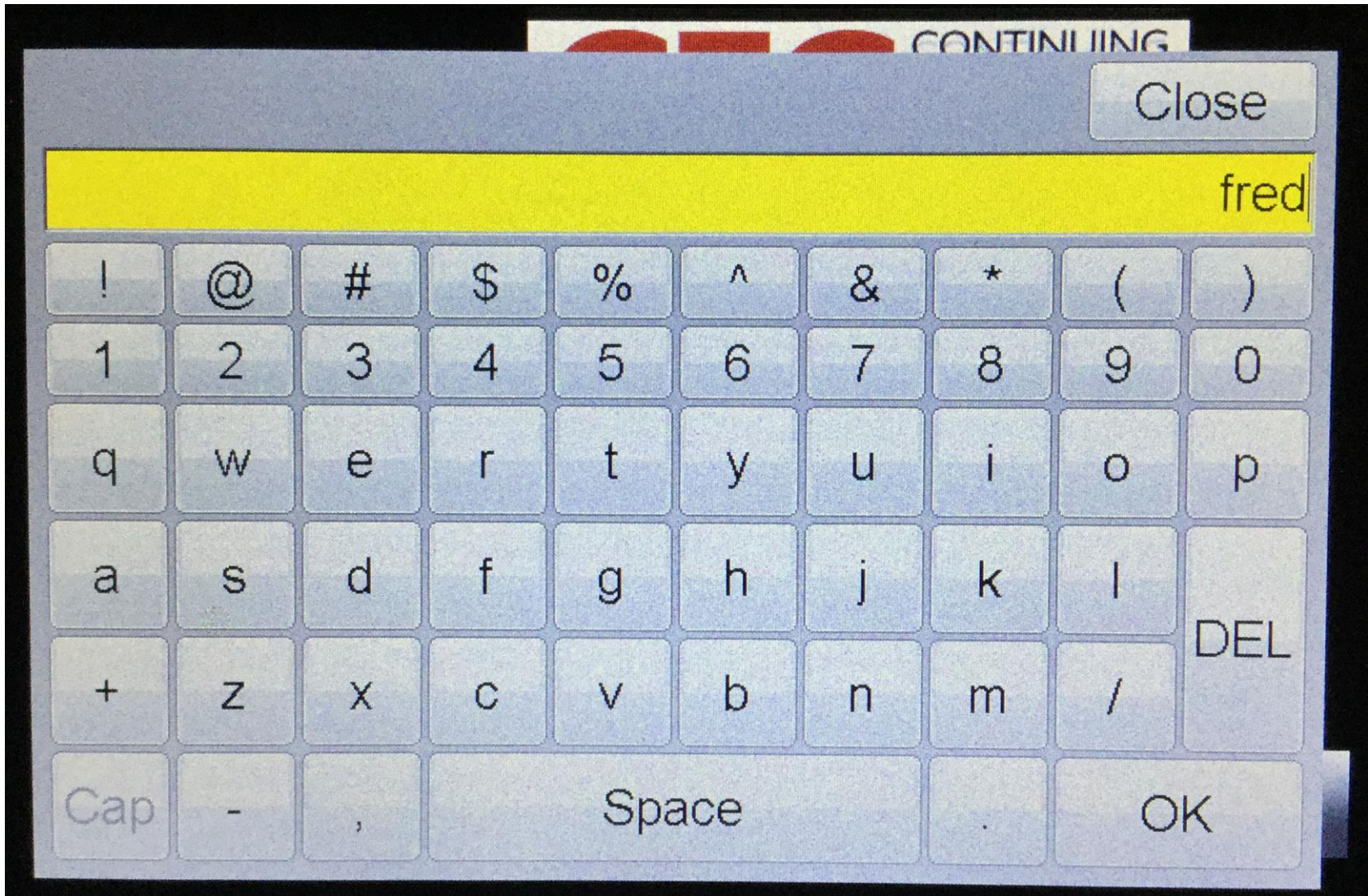
MAIL TO: Mail to p

MAIL FROM: Mail from p

Touch Key
SAVE

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing



Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

CEC CONTINUING
EDUCATION
CENTER

USER NAME fred

PASSWORD

EMAIL DOMAIN

MAIL TO:

MAIL FROM:

SAVE

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

```
538 //*****
539 /** CHECK RECEIVE BUFFER
540 //*****
541 uint8_t getLCDpacket(void)
542 {
543     uint8_t rc,i,j;
544
545     rc = 0;
546     if(CharInRing())
547     {
548         rxBufLCD[0] = readring();
549         if(rxBufLCD[0] == 0xAA)
550         {
551             HAL_Delay(20);
552             idxBufLCD = 1;
553             do{
554                 rxBufLCD[idxBufLCD++] = readring();
555                 if(rxBufLCD[idxBufLCD-1] == 0xCC)
556                 {
557                     rxBufLCD[idxBufLCD++] = readring();
558                     if(rxBufLCD[idxBufLCD-1] == 0x33)
559                     {
560                         rxBufLCD[idxBufLCD++] = readring();
561                         if(rxBufLCD[idxBufLCD-1] == 0xC3)
562                         {
563                             rxBufLCD[idxBufLCD++] = readring();
564                             if(rxBufLCD[idxBufLCD-1] == 0x3C)
565                             {
566                                 break;
567                             }
568                         }
569                     }
570                 }
571             }while (CharInRing());
572         }
```


Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

Watch 1

Name	Value	Type
USART1_RxBuf	0x20000494 USART1_R...	unsigned char[256]
[0]	0x00	unsigned char
[1]	0xAA 'à'	unsigned char
[2]	0x78 'x'	unsigned char
[3]	0x00	unsigned char
[4]	0x00	unsigned char
[5]	0x00	unsigned char
[6]	0xCC 'ì'	unsigned char
[7]	0x33 '3'	unsigned char
[8]	0xC3 'Ã'	unsigned char
[9]	0x3C '<'	unsigned char
[10]	0xAA 'à'	unsigned char
[11]	0x77 'w'	unsigned char
[12]	0x00	unsigned char
[13]	0x00	unsigned char
[14]	0x00	unsigned char
[15]	0x80 '€'	unsigned char
[16]	0x66 'f'	unsigned char
[17]	0x72 'r'	unsigned char
[18]	0x65 'e'	unsigned char
[19]	0x64 'd'	unsigned char
[20]	0x00	unsigned char
[21]	0xCC 'ì'	unsigned char
[22]	0x33 '3'	unsigned char
[23]	0xC3 'Ã'	unsigned char
[24]	0x3C '<'	unsigned char
[25]	0x00	unsigned char

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

Touch Key ID Response code (0x78):

Seq.	Content	Byte in Hex	Descriptions
1 st	Header	0xAA	Communication packet header
2 nd	Command	0x78	Touched release Key_ID defined by TOPWAY TML Graphic Editor will be response to host
3 rd	Page_ID	Page_IDh	Page_ID = the touch key in page(16bit binary value)
4 th		Page_IDl	
5 th	Y coordinate	Key_ID	Key_ID (8bit binary value)
6 th	Tail	0xCC	Communication packet tail
7 th		0x33	
8 th		0xC3	
9 th		0x3C	

Name	Value	Type
rxBufLCD	0x20000594 rxBufLCD[...]	unsigned char[256]
[0]	0xAA 'ª'	unsigned char
[1]	0x78 'x'	unsigned char
[2]	0x00	unsigned char
[3]	0x00	unsigned char
[4]	0x00	unsigned char
[5]	0xCC 'ì'	unsigned char
[6]	0x33 '3'	unsigned char
[7]	0xC3 'Ã'	unsigned char
[8]	0x3C '<'	unsigned char
[9]	0x00	unsigned char

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

```
634 //switch(rxBufLCD[1])
635 case 0x78:
636     resp.cmd = rxBufLCD[1];
637     resp.pageIDh = rxBufLCD[2];
638     resp.pageIDl = rxBufLCD[3];
639     resp.keyID = rxBufLCD[4];
640     pageAddr = make16(resp.pageIDh, resp.pageIDl);
641     switch(pageAddr)
642     {
643     case displaypage0000:
644         switch(resp.keyID)
645         {
646         case tksave_page0000:
647             if(lenusrname > 0)
648             {
649                 pageWR(eepageusrname, emaildata.uname, &lenusrname);
650                 pageRD(eepageusrname, dst64, &lenusrname);
651             }
652             if(lenpasswd > 0)
653             {
654                 pageWR(eepagepasswd, emaildata.pword, &lenpasswd);
655                 pageRD(eepagepasswd, dst64, &lenpasswd);
656             }
657             if(lenedomain > 0)
658             {
659                 pageWR(eepagedomain, emaildata.domain, &lenedomain);
660                 pageRD(eepagedomain, dst64, &lenedomain);
661             }
662             if(lenmailto > 0)
663             {
664                 pageWR(eepagemailto, emaildata.emto, &lenmailto);
665                 pageRD(eepagemailto, dst64, &lenmailto);
666             }
667             if(lenmailfrm > 0)
668             {
669                 pageWR(eepagemailfrm, emaildata.emfrm, &lenmailfrm);
670                 pageRD(eepagemailfrm, dst64, &lenmailfrm);
671             }
672             break;
673         }
674     }
675     break;
```

```
/**
**/*****
**/ DISPLAY PAGE DEFINITIONS
**/*****
#define displaypage0000 0x0000
**/*****
**/ DISPLAY TOUCH KEY ID DEFINITIONS
**/*****
#define tksave_page0000 0x05
```

```
88 typedef struct{
89     uint8_t  uname[128];
90     uint8_t  pword[128];
91     uint8_t  domain[128];
92     uint8_t  emto[128];
93     uint8_t  emfrm[128];
94 }EMAILDATA;
95 EMAILDATA emaildata;
96
97 typedef struct{
98     uint8_t  cmd;
99     uint8_t  pageIDh;
100    uint8_t  pageIDl;
101    uint8_t  keyID;
102    uint8_t  vpAddr3;
103    uint8_t  vpAddr2;
104    uint8_t  vpAddr1;
105    uint8_t  vpAddr0;
106    uint32_t vpAddr;
107    //uint8_t  vpN16h;
108    //uint8_t  vpN16l;
109    //uint16_t vpN16;
110    //uint8_t  vpNascii[5];
111    uint8_t  vpStr[128];
112 }TOUCHRESPONSE;
113 TOUCHRESPONSE resp;
```


Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

Touch Key VP_ADD+VP_Value Response code (0x77):

Seq.	Content	Byte in Hex	Descriptions
1 st	Header	0xAA	Communication packet header
2 nd	Command	0x77	Touch Key VP_ADD+VP_Value Response code
3 rd	VP_ADD	Addr3 (MSB)	VP Address 0x080000 ~ 0x08FFFF:VP_N16 Address 0x020000 ~ 0x02FFFF:VP_N32 Address
4 th		Addr2	
5 th		Addr1	
6 th		Addr0(LSB)	
7 th	Data	:	
:		:	
:		:	
:		:	
:	Tail	0xCC	
:		0x33	
:		0xC3	
:		0x3C	

Name	Value	Type
rxBufLCD	0x20000594 rxBufLCD[...]	unsigned char[256]
[0]	0xAA 'a'	unsigned char
[1]	0x77 'w'	unsigned char
[2]	0x00	unsigned char
[3]	0x00	unsigned char
[4]	0x00	unsigned char
[5]	0x80 '€'	unsigned char
[6]	0x66 'f'	unsigned char
[7]	0x72 'r'	unsigned char
[8]	0x65 'e'	unsigned char
[9]	0x64 'd'	unsigned char
[10]	0x00	unsigned char
[11]	0xCC 'ì'	unsigned char
[12]	0x33 '3'	unsigned char
[13]	0xC3 'Ã'	unsigned char
[14]	0x3C '<'	unsigned char
[15]	0x00	unsigned char

```

//*****
//* DISPLAY STRING MEMORY ADDRESSES
//*****
#define username      0x00000080
#define passwd       0x00000100
#define edomain      0x00000180
#define mailto       0x00000200
#define mailfrom     0x00000280
    
```

Presented by:

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

```
switch(rxBufLCD[1])
{
    case 0x77:
        resp.cmd = rxBufLCD[1];
        resp.vpAddr3 = rxBufLCD[2];
        resp.vpAddr2 = rxBufLCD[3];
        resp.vpAddr1 = rxBufLCD[4];
        resp.vpAddr0 = rxBufLCD[5];
        resp.vpAddr = make32(resp.vpAddr3, resp.vpAddr2, resp.vpAddr1, resp.vpAddr0);
        switch (resp.vpAddr)
        {
            case username:
                i = 0;
                j = 6;
                lenusrname = 0;
                do{
                    emaildata.uname[i++] = rxBufLCD[j++];
                    lenusrname++;
                }while(rxBufLCD[j-1] != 0x00);
                break;
            case passwd:
                i = 0;
                j = 6;
                lenpasswd = 0;
                do{
                    emaildata.pword[i++] = rxBufLCD[j++];
                    lenpasswd++;
                }while(rxBufLCD[j-1] != 0x00);
                break;
```

```
/**
 * DISPLAY STRING MEMORY ADDRESSES
 */
#define username    0x00000080
#define passwd     0x00000100
#define edomain    0x00000180
#define mailto     0x00000200
#define mailfrom   0x00000280
```

```
88 typedef struct{
89     uint8_t  uname[128];
90     uint8_t  pword[128];
91     uint8_t  domain[128];
92     uint8_t  emto[128];
93     uint8_t  emfrm[128];
94 }EMAILDATA;
95 EMAILDATA emaildata;
96
97 typedef struct{
98     uint8_t  cmd;
99     uint8_t  pageIDh;
100    uint8_t  pageIDl;
101    uint8_t  keyID;
102    uint8_t  vpAddr3;
103    uint8_t  vpAddr2;
104    uint8_t  vpAddr1;
105    uint8_t  vpAddr0;
106    uint32_t vpAddr;
107    //uint8_t  vpN16h;
108    //uint8_t  vpN16l;
109    //uint16_t vpN16;
110    //uint8_t  vpNascii[5];
111    uint8_t  vpStr[128];
112 }TOUCHRESPONSE;
113 TOUCHRESPONSE resp;
```

Presented by:

Essential Coding Techniques for Hardware Engineers

Touch Display - The Logical Interface: Packet Processing

Watch 1

Name	Value	Type
lenusname	0x05	unsigned char
emaildata	0x2000017C &emaildata	struct <untagged>
uname	0x2000017C &emailda...	unsigned char[128]
[0]	0x66 'f'	unsigned char
[1]	0x72 'r'	unsigned char
[2]	0x65 'e'	unsigned char
[3]	0x64 'd'	unsigned char
[4]	0x00	unsigned char

CEC CONTINUING EDUCATION CENTER

USER NAME fred

PASSWORD

EMAIL DOMAIN

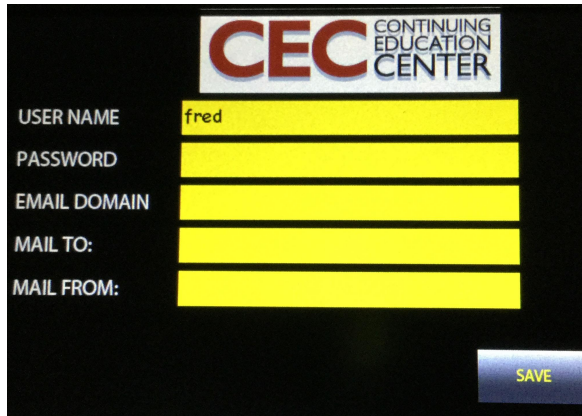
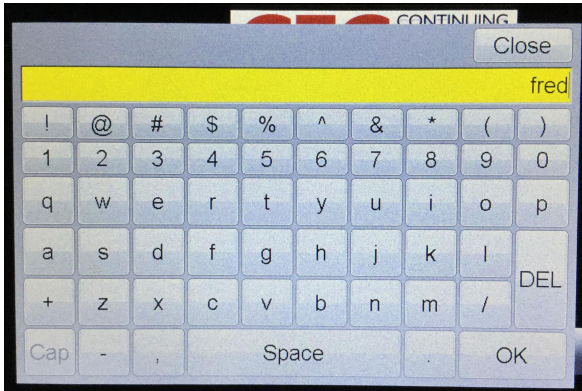
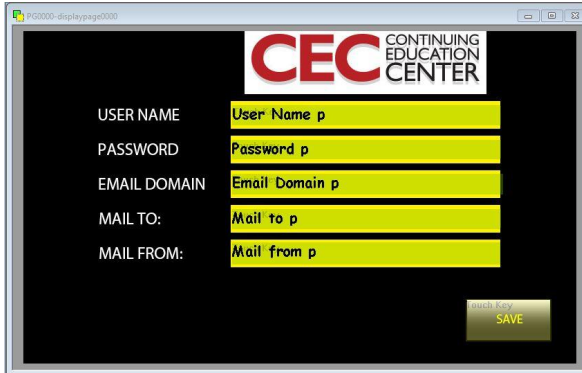
MAIL TO:

MAIL FROM:

SAVE

Essential Coding Techniques for Hardware Engineers

Day 2 Summary



Name	Value	Type
USART1_RxBuf	0x20000494 USART1_R...	unsigned char[256]
[0]	0x00	unsigned char

Name	Value	Type
lenusername	0x05	unsigned char
emaildata	0x2000017C &emaildata	struct <untagged>
uname	0x2000017C &emailda...	unsigned char[128]
[0]	0x66 'f'	unsigned char
[1]	0x72 'r'	unsigned char
[2]	0x65 'e'	unsigned char
[3]	0x64 'd'	unsigned char
[4]	0x00	unsigned char
[12]	0x00	unsigned char
[13]	0x00	unsigned char
[14]	0x00	unsigned char
[15]	0x80 '€'	unsigned char
[16]	0x66 'f'	unsigned char
[17]	0x72 'r'	unsigned char
[18]	0x65 'e'	unsigned char
[19]	0x64 'd'	unsigned char
[20]	0x00	unsigned char
[21]	0xCC 'ì'	unsigned char
[22]	0x33 '3'	unsigned char
[23]	0xC3 'Ä'	unsigned char
[24]	0x3C '<'	unsigned char
[25]	0x00	unsigned char

Essential Coding Techniques for Hardware Engineers

A Peek At What's To Come

