

U-UART - USB UltraBaud Data Transfer IC with RS232 / RS422 and CPU I/F Options**FEATURES**

- Single Chip Multi-Function Data Transfer Solution
 - RS232 link from 300 baud to 920K baud
 - RS422/RS485 Link to 2000K baud
 - 384 byte receive buffer / 128 byte transmit buffer for high data throughput
 - Full hardware assisted or X-On/X-Off handshaking
 - Support for Event Characters and Line Break condition
 - Auto Transmit Buffer control for RS485
 - Compact 32 pin (7mm x 7mm) MQFP package
 - Integrated 6MHz - 48MHz Clock Multiplier aids FCC and CE compliance
 - Integrated 3.3v Regulator – No External Regulator Required
 - 4.4v .. 5.25v Single Supply Operation
 - UHCI / OHCI Compliant
 - USB 1.1 Specification Compliant
 - USB VID, PID, Serial Number and Product Description Strings in external E2PROM.
- Virtual COM Port Drivers for –
- Windows 98 and Windows 98 SE
 - Windows 2000
 - Windows Millennium **
 - Apple iMAC **
 - Linux **
- Application Areas
 - USB ISDN and ADSL Modems
 - USB 56k / V90 Modems
 - USB ⇔ PDA Interface Cables
 - USB ⇔ RS232 Converters / Cables
 - USB ⇔ RS422 / RS485 High Speed Industrial Links
 - USB Digital Cameras
 - USB I/F for MP3 players
 - Ultra-high performance Serial Port for legacy - free PC system boards / Easy PC's
 - USB Instrumentation
 - USB ⇔ USB data transfer cables
 - USB ⇔ USB null-modem cables
 - USB ⇔ Serial Bar Code Readers

Note ** = Currently in development

GENERAL DESCRIPTION

The FT8U232AM is a cost-effective single chip USB UART (U-UART) solution for transferring serial data over USB. With data transfer rates of up to 920k baud (RS232) and 2,000 k baud (RS422 / RS485) , the FT8U232AM significantly raises the performance level above that of traditional ISA and PCI based UART solutions whilst offering true plug and play and easy interfacing through it's USB interface.

Its flexible architecture allows this IC to be used in many different application areas - USB modems, Legacy RS232 ⇔ USB Converter cables, USB interface cables for PDA's , Bar Code Scanners , RS422 data links and instrumentation – in fact almost any equipment that previously used a slow RS232 link for communication.

Virtual COM port drivers are available for the FT8U232AM for Windows '98, Windows 98 SE and Windows 2000. Drivers for other operating systems are currently under development.

Figure 1 – FT8U232AM Block Diagram (Simplified)

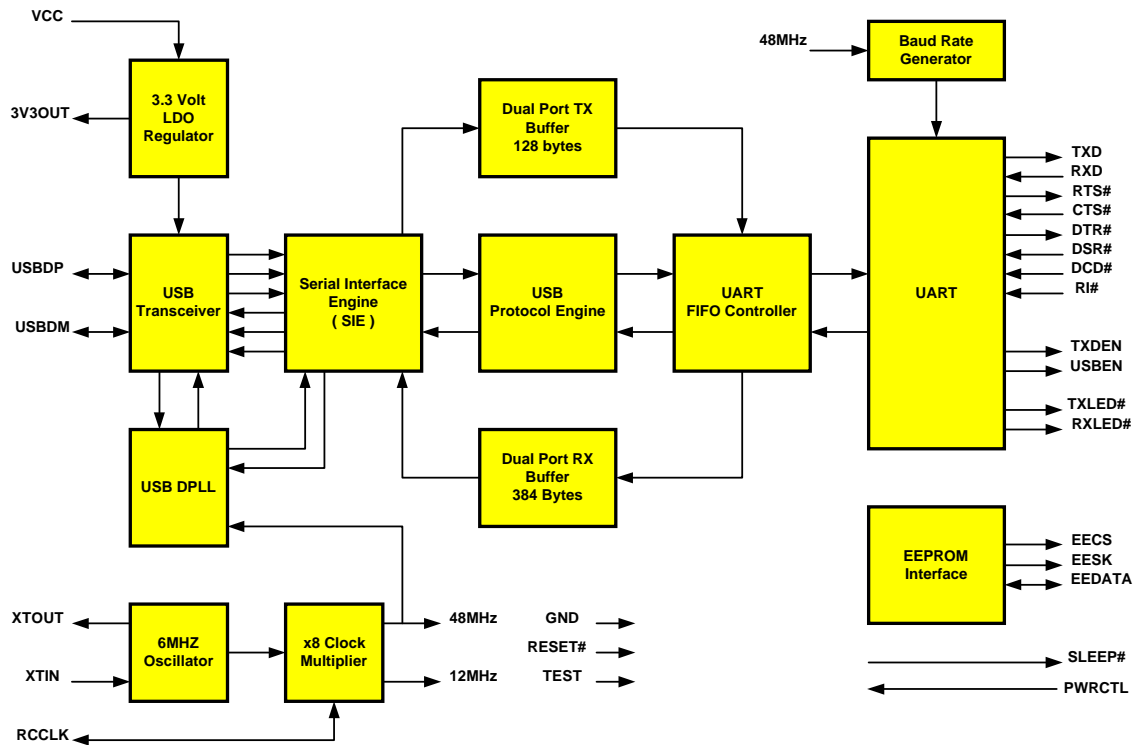
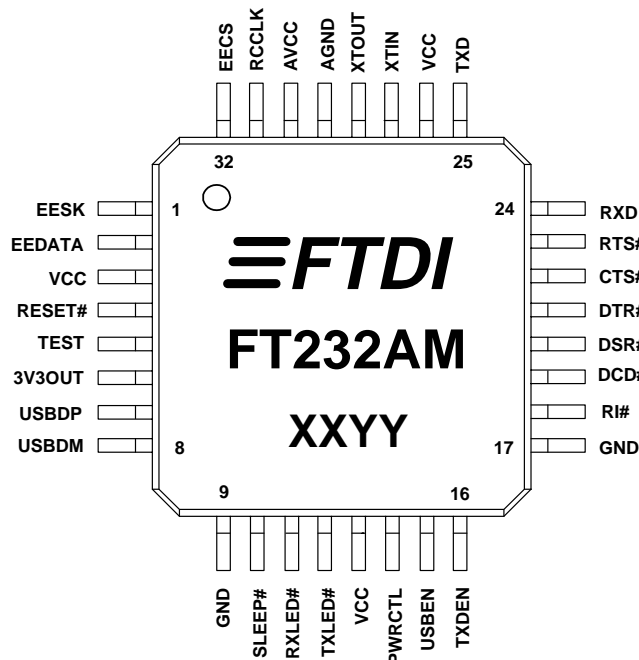


Figure 2 – FT8U232AM I.C. Pinout



FT8U232AM - FUNCTIONAL BLOCK DESCRIPTION

- **3.3V LDO Regulator**

The 3.3V LDO Regulator generates the 3.3 volt reference voltage for driving the USB transceiver cell output buffers. It requires an external decoupling capacitor to be attached to the 3V3OUT regulator output pin.

- **USB Transceiver**

The USB Transceiver Cell provides the USB 1.1 full-speed physical interface to the USB cable. The output drivers provide 3.3 volt level slew rate control signalling, whilst a differential receiver and two single ended receivers provide USB data in, SEO and USB Reset condition detection.

- **USB DPLL**

The USB DPLL cell locks on to the incoming NRZI USB data and provides separate recovered clock and data signals to the SIE block.

- **6MHz Oscillator**

The 6MHz Oscillator cell generates a 6MHz reference clock input to the X8 Clock multiplier from an external 6MHz crystal or ceramic resonator.

- **X8 Clock Multiplier**

The X8 Clock Multiplier takes the 6MHz input from the Oscillator cell and generates a 12MHz reference clock for the SIE, USB Protocol Engine and UART FIFO controller blocks. It also generates a 48MHz reference clock for the USB DPPL and the Baud Rate Generator blocks.

- **Serial Interface Engine (SIE)**

The Serial Interface Engine (SIE) block performs the Parallel to Serial and Serial to Parallel conversion of the USB data. In accordance to the USB 1.1 specification, it performs bit stuffing / un-stuffing and CRC5 / CRC16 generation / checking on the USB data stream.

- **USB Protocol Engine**

The USB Protocol Engine manages the data stream from the device USB control endpoint. It handles the low level USB protocol (Chapter 9) requests generated by the USB host controller and the commands for controlling the functional parameters of the UART.

- **Dual Port TX Buffer (128 bytes)**

Data from the USB data out endpoint is stored in the Dual Port TX buffer and removed from the buffer to the UART transmit register under control of the UART FIFO controller.

- **Dual Port RX Buffer (384 bytes)**

Data from the UART receive register is stored in the Dual Port RX buffer prior to being removed by the SIE on a USB request for data from the device data in endpoint.

- **UART FIFO Controller**

The UART FIFO controller handles the transfer of data between the Dual Port RX and TX buffers and the UART transmit and receive registers.

- **UART**

The UART performs asynchronous 7 / 8 bit Parallel to Serial and Serial to Parallel conversion of the data on the RS232 (RS422 and RS485) interface. Control signals supported by the UART include RTS, CTS, DSR , DTR, DCD and RI. The UART provides a transmitter enable control signal (TXDEN) to assist with interfacing to RS485 transceivers.

- **Baud Rate Generator**

The Baud Rate Generator provides a x16 clock input to the UART from the 48MHz reference clock and consists of a 14 bit prescaler and 2 register bits which provide fine tuning of the baud rate (e.g. to divide by 2.5). This determines the Baud Rate of the UART which is programmable from 300 baud to 2 million baud. For more details please contact FTDI.

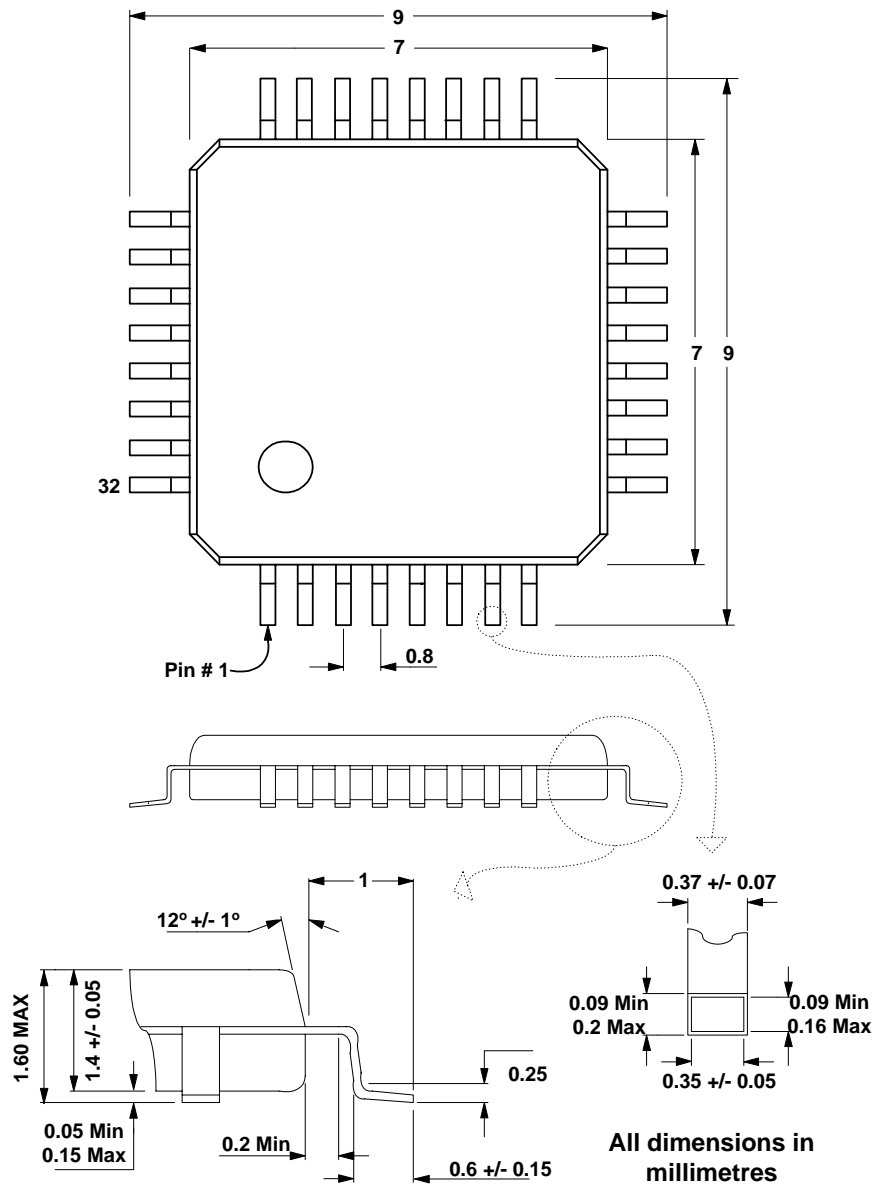
- **EEPROM Interface**

Though the FT8U232AM will work without the optional EEPROM, an external 93C46 EEPROM can be used to customise the USB VID, PID, Serial Number and Strings of the FT8U232AM for OEM applications. The EEPROM is also required for applications where multiple FT8U232AM's are connected to a single PC as the drivers rely on a unique serial number for each device to bind a unique virtual COM port to each individual device.

Table 1 - FT8U232AM - PINOUT DESCRIPTION

Pin #	Signal	Type	Description
7	USBDP	I/O	USB Data Signal Plus – Requires 1.5k pull-up to 3V3OUT
8	USBDM	I/O	USB Data Signal Minus
6	3V3OUT	OUT	3.3 volt Output from integrated regulator
27	XTIN	IN	Input to 6MHz Crystal Oscillator Cell
28	XTOUT	OUT	Output from 6MHz Crystal Oscillator Cell
31	RCCLK	I/O	RC timer – used to guarantee clock stability on exiting sleep mode
4	RESET#	IN	Resets entire device using external RC network
32	EECS	I/O	Optional EEPROM – Chip Select
1	EESK	I/O	Optional EEPROM – Clock
2	EEDATA	I/O	Optional EEPROM – Data I/O
5	TEST	IN	Puts device in i.c. test mode – must be tied to GND
25	TXD	OUT	UART – Transmit Data Output
24	RXD	IN	UART – Receive Data Input
23	RTS#	OUT	UART – Request To Send Control Output
22	CTS#	IN	UART – Clear To Send Control Input
21	DTR#	OUT	UART – Data Terminal Ready Control Output
20	DSR#	IN	UART – Data Set Ready Control Input
19	DCD#	IN	UART – Data Carrier Detect Control Input
18	RI#	IN	UART – Ring Indicator Control Input
16	TXDEN	OUT	UART – Enable Transmit Data for RS485
15	USBEN	OUT	USB Enabled – High after device is configured via USB
14	PWRCTL	IN	Bus Powered – Tie Low / Self Powered – Tie High
12	TXLED#	O.C.	LED Drive - Pulses Low when Transmitting Data via USB
11	RXLED#	O.C.	LED Drive - Pulses Low when Receiving Data via USB
10	SLEEP#	OUT	Goes Low during USB Suspend Mode
3,13,26	VCC	PWR	Device - +4.4 volt to +5.25 volt Power Supply Pins
9,17	GND	PWR	Device – Ground Supply Pins
30	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier
29	AGND	PWR	Device - Analog Ground Supply for the internal x8 clock multiplier

Figure 3. FT8U232AM - PACKAGE DESCRIPTION – QFP 7mm x 7mm



Absolute Maximum Ratings

Storage Temperature	-65°C to + 150°C
Ambient Temperature (Power Applied).....	0°C to + 70°C
VCC Supply Voltage	-0.5v to +6.00v
DC Input Voltage - Inputs	-0.5v to VCC + 0.5v
DC Input Voltage - High Impedance Bidirectionals	-0.5v to VCC + 0.5v
DC Output Current – Outputs	24mA
DC Output Current – Low Impedance Bidirectionals	24mA
Power Dissipation	500mW

DC Characteristics (Ambient Temperature = 0 .. 70 Degrees C)

	<i>Description</i>	<i>Min</i>	<i>Max</i>	<i>Units</i>	<i>Conditions</i>
VCC	Operating Supply Voltage	4.4	5.25	v	
lcc1	Operating Supply Current		50	mA	Normal Operation
lcc2	Operating Supply Current		250	uA	USB Suspend
loh1	Digital IO Pins Source Current	4		mA	Voh = VCC – 0.5v
lol1	Digital IO Pins Sink Current	8		mA	Vol = + 0.5v
Voh1	Input Voltage Threshold (Low)		0.6	v	
Vol1	Input Voltage Threshold (High)	2.7		v	
VDif	USB Differential Input Sensitivity	0.2		v	
VCom	USB Differential Common Mode	0.8	2.5	v	
URxt	USB Single Ended Rx Threshold	0.8	2.0	v	
UVh	USB IO Pins Static Output (Low)		0.3v		RI = 1.5k to 3.6v
UVI	USB IO Pins Static Output (High)	2.8			RI = 15k to GND

Disclaimer

© Future Technology Devices International Limited – 1999, 2000

Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder.

This product and its documentation are supplied on an as is basis and no warranty as to their suitability for any particular purpose is either made or implied.

Future Technology Devices International Ltd. will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected.

This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury.

This document provides preliminary information that may be subject to change without notice.

Contact Information

Future Technology Devices Intl. Limited	Telephone :	+44 (0) 141 353 2565
St. George's Studios	Fax :	+44 (0) 141 353 2656
93/97 St. George's Road	Email :	support@ftdi.co.uk
Glasgow G3 6JA, UK	Internet :	http://www.ftdi.co.uk

Agents and Sales Representatives

Please visit our internet site for the latest contact details of our Agents and Sales Representatives world-wide.

Appendix A

USB Device Descriptors

USB Device Descriptors

Note: **E** - replaced by E2Rom Value, **C** - modified by configuration option

```
                                { * device descriptor *}
LABEL : Device_Des;

    0010 12      Val : Device_Len;{length of this descriptor in bytes}
    0011 01      Val : $01;{Device descriptor type}
    0012 10 01   Val : $10,$01;{USB Spec rev 1.10}
    0014 00      Val : $00;{Device class ?}
    0015 00      Val : $00;{Device subclass ?}
    0016 00      Val : $00;{Device protocol ?}
    0017 08      Val : Max_Length;{maximum packet size}

LABEL : Device_Des_Vendor;

E 0018 03 04   Val : $03,$04;{Vendor ID FTDI}
E 001A 01 60   Val : $01,$60;{product number 6001}

LABEL : Device_Des_Vendor_End;

E 001C 00 02   Val : $00,$02;{device release number 02.00}
    001E 01     Val : $01;{index of string descriptor describing manufacturer}
    001F 02     Val : $02;{index of string descriptor describing product}
    0020 03     Val : $03;{index of string descriptor describing serial number}
    0021 01     Val : $01;{number of possible configurations}

{ * end of device descriptor *}
LABEL : Device_Des_End;

LABEL : Config_Des;
{ * configuration descriptor *}

    0022 09      Val : $09;{length of this descriptor in bytes}
    0023 02      Val : $02;{Configuration descriptor}
    0024 20 00   Val : Config_Len,$00;{length of data returned for all things}
    0026 01      Val : $01;{number of interfaces supported by this configuration}
    0027 01      Val : $01;{configuration value}
    0028 00      Val : $00;{index of string descriptor describing this configuration}
EC 0029 80     Val : 10000000b;{configured as bus powered and not remote wakeup}
E 002A 2D      Val : 45;{maximum power in 2 mA ie 90mA for now}

{ * end of configuration descriptor *}

LABEL : Interface_Des;
{ * interface descriptor *}

    002B 09      Val : $09;{length of this descriptor in bytes}
    002C 04      Val : $04;{Interface descriptor}
    002D 00      Val : $00;{interface number}
    002E 00      Val : $00;{alternate setting}
    002F 02      Val : $02;{number of endpoints excluding 0 = 1}
    0030 FF      Val : $ff;{class code}
    0031 FF      Val : $ff;{subclass}
    0032 FF      Val : $ff;{protocol code}
    0033 02      Val : $02;{index of string descriptor describing this interface}

{ * end of interface descriptor *}

LABEL : Interface_Des_End;

LABEL : Endpoint_Des;

LABEL : Endpoint3_Des_End;

    0034 07      Val : $07;{length of this descriptor in bytes}
    0035 05      Val : $05;{End point descriptor}
    0036 81      Val : 10000001b;{in endpoint at address 1}
    0037 02      Val : 00000010b;{attribute as bulk}
    0038 40 00   Val : 64,$00;{maximum packet size}
    003A 00      Val : $00;{interval for polling endpoint for data transfers}
```

```

LABEL : Endpoint3_Des; { * end point descriptor * }

003B 07      Val : $07; {length of this descriptor in bytes}
003C 05      Val : $05; {End point descriptor}
003D 02      Val : 00000010b; {out endpoint at address 2}
003E 02      Val : 00000010b; {attribute as bulk}
003F 40 00   Val : 64,$00; {maximum packet size}
0041 00      Val : $00; {interval for polling endpoint for data transfers}

```

```

LABEL : Endpoint_Des_End;
LABEL : Config_Des_End;

```

```

LABEL : Str0_Des;

```

```

0042 04      Val : Str0_Len;          {length of string descriptor}
0043 03      Val : $03;              {type string}
0044 09 04   Val : $09,$04;         {language ID 0009 English}
LABEL : Str0_Des_End;

```

```

LABEL : Str1_Des;

```

```

E 0046 0A      Val : Str1_Len;          {length of string descriptor}
E 0047 03      Val : $03;              {type string}
E 0048 46 00   Val : 'F', $00;
E 004A 54 00   Val : 'T', $00;
E 004C 44 00   Val : 'D', $00;
E 004E 49 00   Val : 'I', $00;
LABEL : Str1_Des_End;

```

```

LABEL : Str2_Des;

```

```

E 0050 1E      Val : Str2_Len;          {length of string descriptor}
E 0051 03      Val : $03;              {type string}
E 0052 55 00   Val : 'U', $00;
E 0054 53 00   Val : 'S', $00;
E 0056 42 00   Val : 'B', $00;
E 0058 20 00   Val : ' ', $00;
E 005A 3C 00   Val : '<', $00;
E 005C 2D 00   Val : '-', $00;
E 005E 3E 00   Val : '>', $00;
E 0060 20 00   Val : ' ', $00;
E 0062 53 00   Val : 'S', $00;
E 0064 65 00   Val : 'e', $00;
E 0066 72 00   Val : 'r', $00;
E 0068 69 00   Val : 'i', $00;
E 006A 61 00   Val : 'a', $00;
E 006C 6C 00   Val : 'l', $00;
LABEL : Str2_Des_End;

```

```

LABEL : Str3_Des;

```

```

E 006E 12      Val : Str3_Len;          {serial number string}
E 006F 03      Val : $03;              {type string}
E 0070 31 00   Val : '1', 00;
E 0072 32 00   Val : '2', 00;
E 0074 33 00   Val : '3', 00;
E 0076 34 00   Val : '4', 00;
E 0078 35 00   Val : '5', 00;
E 007A 36 00   Val : '6', 00;
E 007C 37 00   Val : '7', 00;
E 007E 38 00   Val : '8', 00;
LABEL : Str3_Des_End;

```

Appendix B

EEPROM Data Structure

E2Rom Data example

```
0000 00 00 Val : $00,$00;{Configuration value}
0002 03 04 Val : $03,$04;{Vendor ID FTDI}
0004 01 60 Val : $01,$60;{product number 6001}
0006 00 02 Val : $00,$02;{device release number}
0008 A0 Val : 10100000b; {config descriptor value bus powered and remote wakeup}
0009 2D Val : 45; {max power = value * 2 mA}
000A 00 00 Val : $00,$00;{reserved}
000C 00 00 Val : $00,$00;{reserved}
```

```
000E 94 VAL : PTR_ManStringDes;
000F 0C Val : ManStringDes_Len; {length of string descriptor}
0010 A0 VAL : PTR_ProdStringDes;
0011 34 Val : ProdStringDes_Len; {length of string descriptor}
0012 D4 VAL : PTR_SerStringDes;
0013 12 Val : SerStringDes_Len;
```

LABEL : ManStringDes;

```
0014 0C Val : ManStringDes_Len; {length of string descriptor}
0015 03 Val : $03; {type string}
0016 41 00 Val : 'A', $00;
0018 6E 00 Val : 'n', $00;
001A 64 00 Val : 'd', $00;
001C 79 00 Val : 'y', $00;
001E 73 00 Val : 's', $00;
LABEL : ManStringDes_End;
```

LABEL : ProdStringDes;

```
0020 34 Val : ProdStringDes_Len; {length of string descriptor}
0021 03 Val : $03; {type string}
0022 57 00 Val : 'W', $00;
0024 6F 00 Val : 'o', $00;
0026 6E 00 Val : 'n', $00;
0028 64 00 Val : 'd', $00;
002A 65 00 Val : 'e', $00;
002C 72 00 Val : 'r', $00;
002E 66 00 Val : 'f', $00;
0030 75 00 Val : 'u', $00;
0032 6C 00 Val : 'l', $00;
0034 6C 00 Val : 'l', $00;
0036 20 00 Val : ' ', $00;
0038 55 00 Val : 'U', $00;
003A 53 00 Val : 'S', $00;
003C 42 00 Val : 'B', $00;
003E 20 00 Val : ' ', $00;
0040 3C 00 Val : '<', $00;
0042 2D 00 Val : '-', $00;
0044 3E 00 Val : '>', $00;
0046 20 00 Val : ' ', $00;
0048 53 00 Val : 'S', $00;
004A 65 00 Val : 'e', $00;
004C 72 00 Val : 'r', $00;
004E 69 00 Val : 'i', $00;
0050 61 00 Val : 'a', $00;
0052 6C 00 Val : 'l', $00;
LABEL : ProdStringDes_End;
```

LABEL : SerStringDes;

```
0054 12 Val : SerStringDes_Len;
0055 03 Val : $03; {type string}
0056 32 00 Val : '2', 00;
0058 32 00 Val : '2', 00;
005A 33 00 Val : '3', 00;
005C 34 00 Val : '4', 00;
005E 35 00 Val : '5', 00;
0060 36 00 Val : '6', 00;
0062 37 00 Val : '7', 00;
0064 38 00 Val : '8', 00;
LABEL : SerStringDes_End;
```

```
0066 00 00 Val : $00,$00; {reserved for Checksum}
```