UNIQUE MICRO DESIGN

UMD ProtoLink Architecture

Reference Manual

Technologists and suppliers to Professional systems integrators.

Revision History		
Date	Issue	Comments
30/01/96	1	First Issue, replacing STD8006 and STD8007

Unique Micro Design

UMD ProtoLink Architecture Reference Manual

30 January 1996

Issue 1

Unique Micro Design Pty Ltd
Australia

Tel +61 3 9764 8166
Fax +61 3 9764 8177

UMD reference : DOC-PROTOLINK-RM
UMD part number : 7-6070-001-1

Contents

Diagrams

Definitions

“CHR”	Characters between double quote marks represent ASCII printable characters
$3D	Represent hexadecimal values
<STX>	Represents ASCII control characters.
lsb	Least Significant Byte or Least Significant Bit
msb	Most Significant Byte or Least Significant Bit
EPROM	Electrically Programmable Read Only Memory
EEPROM	Electrically Eraseable Programmable Read Only Memory, used as non volatile memory.
RAM	Random Access Memory (Read/Write)
BCR	Bar Code Reader
Preamble	Character string added to the beginning of another string
Postamble	Character string added to the end of another string
Bar Code Symbology	Describes the various algorithms that apply to the printing of bar codes

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Introduction

1 UMD ProtoLink Architecture Introduction

The *UMD ProtoLink Architecture* specifies the following:

Standard definitions for configuration parameters that can be consistently used across a broad range of products.

A standard command set and peripheral control philosophy.

A standard set of hardware facilities which includes non-volatile memory to hold configuration parameters, a peripheral interface bus, serial ports which provide power for scanners and bar code wand, display, external keyboard, keyboard wedge and magnetic card reader interfaces.

A peripheral interface bus that allows the addition of other modules to the core controller.

In essence, the *UMD ProtoLink Architecture* is a versatile product development system.

2 UMD ProtoLink Architecture Specifications

The *UMD ProtoLink Architecture* defines a standard set of interfaces, namely:

Hardware Specifications

5 pin DIN cash drawer interface

5 pin DIN keyboard interface

7 pin DIN host computer interface

AMP DB9 wand interface

Bidirectional parallel interface

Magnetic card reader mechanism interface

ProtoLink peripheral interface

UMD standard DB9 serial port

VGA monitor interface

The *UMD ProtoLink Architecture* also defines software standards by which interfaces and peripherals are linked and controlled:

Software Specifications

ANSI Terminal Emulation

ProtoLink Command Set

ProtoLink Communication Protocol

ProtoLink Configuration Parameters

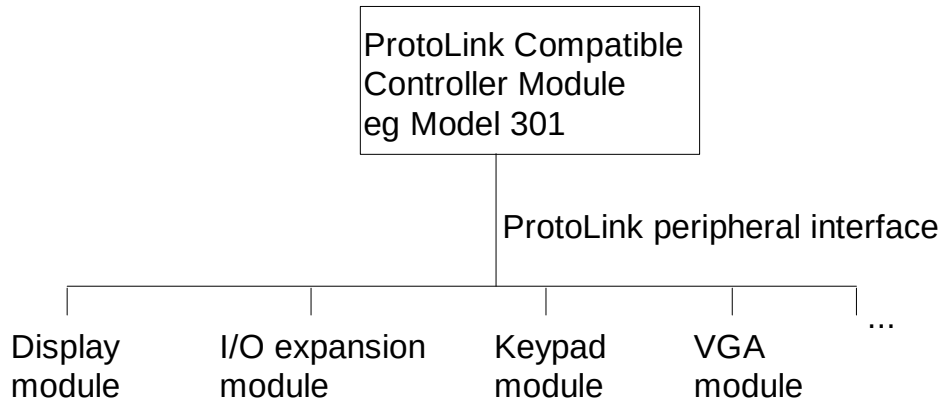
ProtoLink Key Codes

A range of electronic modules have been developed that use this architecture.

The *UMD ProtoLink peripheral interface* is used to interconnect these modules.

By combining enclosure, controller and modules, a number of different products can be assembled. The *UMD ProtoLink* software is embedded in the controller and is used to configure the modules (eg connect keypad to keyboard interface) and customise operation (eg. keypad scan codes, magnetic card reader header information etc).

For example, Unique Micro Design's premier family of *UMD ProtoLink Architecture* products is based on *the UMD Model 301 Controller Module* which is used internally in our range of custom keyboards, wall mount terminals, multi-in/out wedges, peripheral controllers and custom VGA terminals.



3 UMD Model 301 Controller Module

The UMD Model 301 Controller Module is designed to the *UMD ProtoLink Architecture* specifications.

It has the ability to permanently store configuration details in non-volatile memory.

ProtoLink commands are accepted by the controller via its input ports where the incoming stream of data is checked for command sequences. Data streams from the various onboard peripherals and expansion boards can be redirected to other input/output devices via ProtoLink commands.

It has two RS232 serial ports. The pin configuration for the DB9 plugs are a subset of PC/AT standard serial ports with the exception that pin 7 provides 5 Volts to power bar code scanners.

The host computer interface is a bidirectional port which looks like a PC/AT keyboard port. It connects to the keyboard port of a personal computer. This interface also doubles as the power input connector, receiving regulated 5V power from the keyboard connector of a computer or unregulated 6 to 9V from a DC power supply.

The external keyboard interface accepts input from standard PC/AT keyboards.

Keyboard wedging is provided by allowing a PC keyboard connected to the external keyboard interface to communicate with a PC via the host computer keyboard interface.

The bar code decoder interface connects to industry standard digital wands, slot readers and devices that emulate wands.

A hardware reset button is provided to physically reset the microcontroller, forcing it to perform a power up sequence.

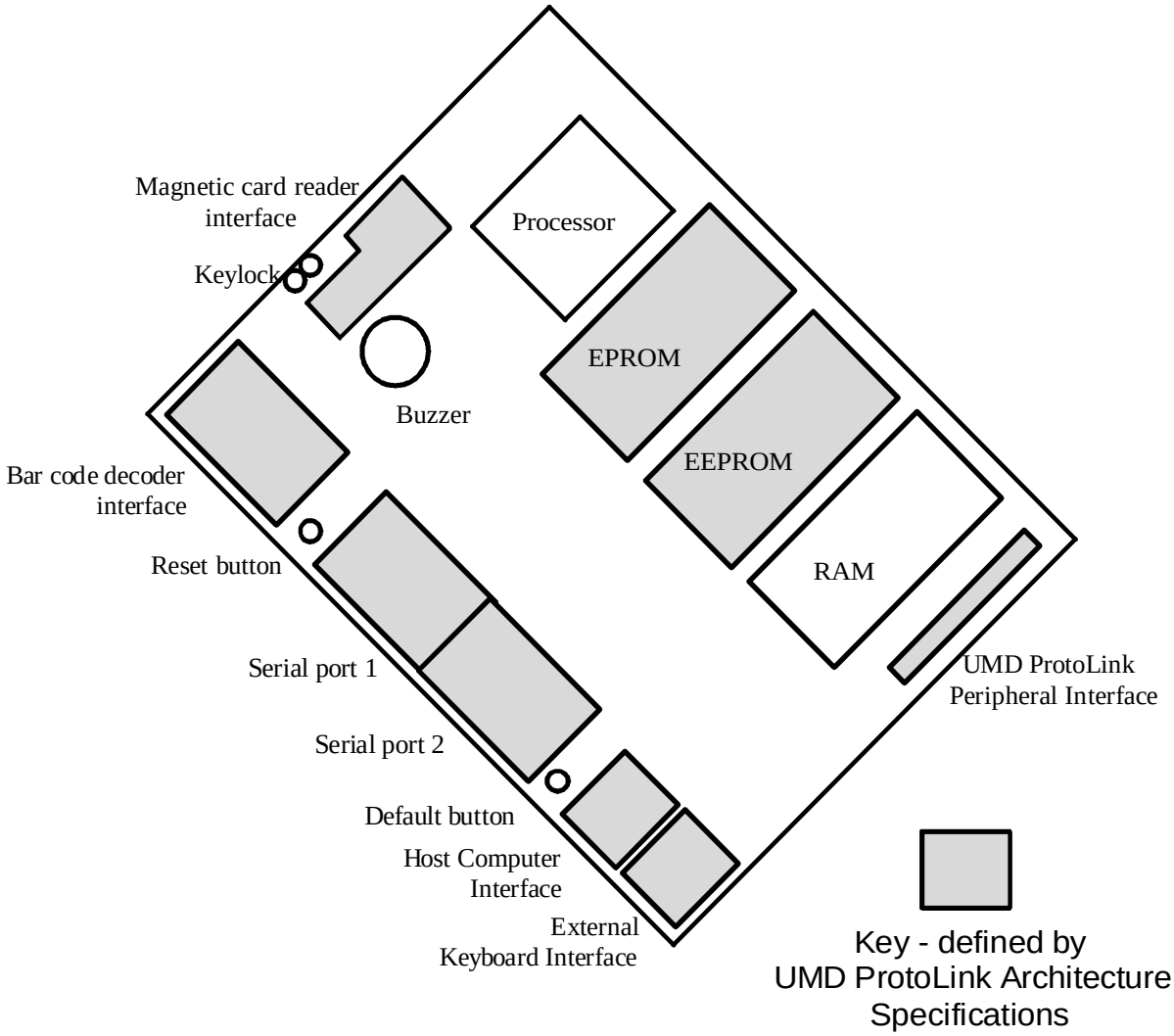
A default button is also provided which is used in conjunction with the reset button to return the non-volatile configuration memory back to factory default settings.

The magnetic card reader interface connects to standard single or dual track MCR mechanisms.

An optional keyboard lock facility disables scanning of keypad modules.

The Model 301 has a ProtoLink peripheral interface for attaching external modules.

There are a number of firmware options available for different applications.



UMD Model 301Controller Module
Physical layout

4 UMD ProtoLink Expansion Modules

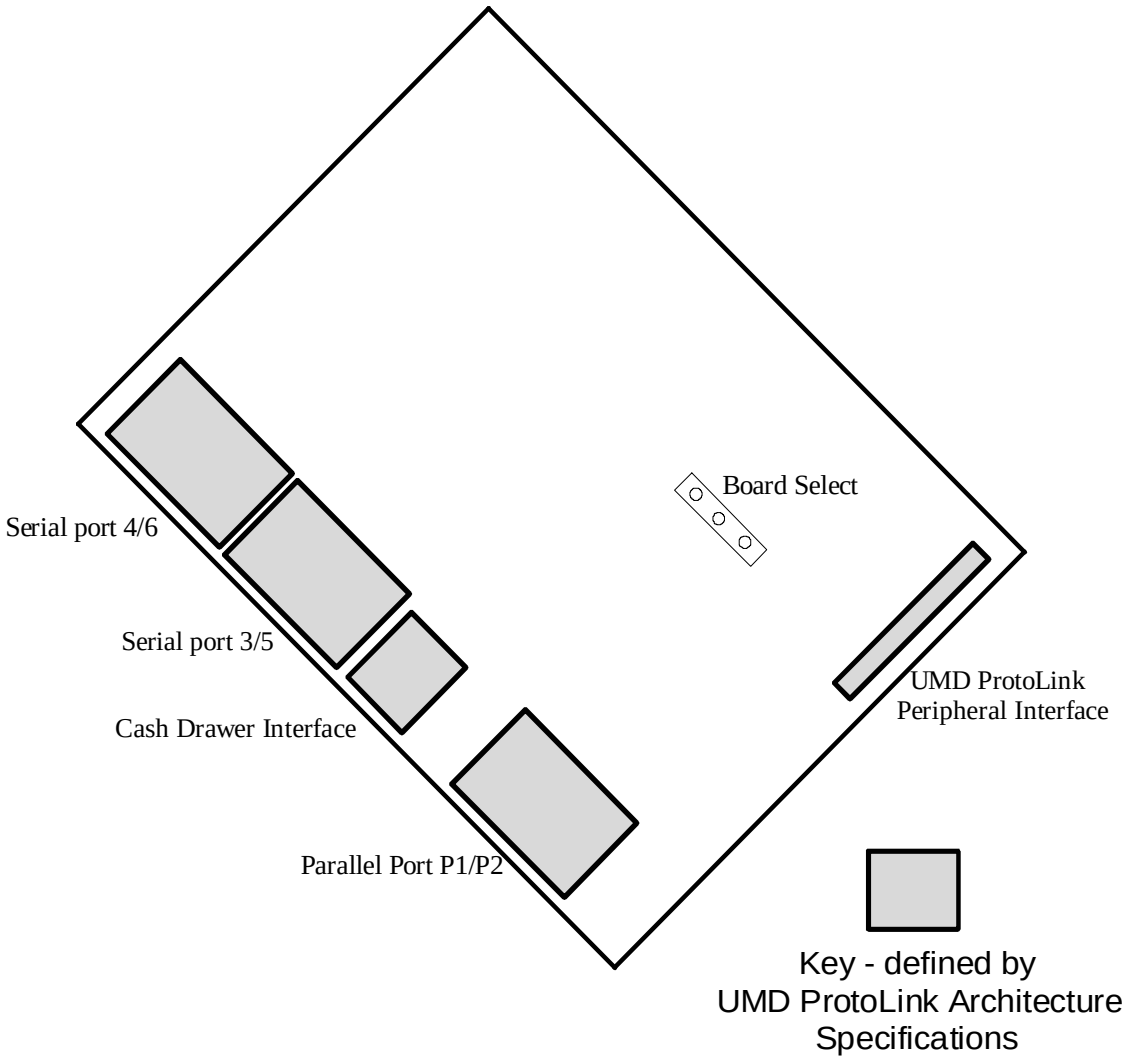
The expansion modules are connected to the controller by the *UMD ProtoLink* peripheral interface, generally via a ribbon cable.

The *UMD Model 301 controller* can accommodate a number of expansion modules which depends on the number and type of modules used.

4.1 I/O Expansion Module

The I/O expansion module provides two full duplex UMD standard DB9 serial ports, a bidirectional parallel port (ie either input or output) and a cash drawer driver interface.

Typical products with the Model 301 and I/O expansion module combination are the UMD Model 221 Peripheral Controller and the UMD Model 264-I Custom Keyboard with I/O expansion.



UMD Model 301Expanded I/O Module
Physical layout

4.2 Display Module

The liquid crystal display (LCD) module provides a backlit character based display.

Currently these displays are available in 2x16 and 2x40 formats. They are commonly used in conjunction with keypad modules to create small serial terminals and industrial control panels.

Typical products with the Model 301 and display module combination are the UMD Model 211 Custom LCD Terminal and the UMD Model 330 Industrial Wall Mount Terminal.

4.3 Keypad Module.

Each keypad module provides a matrix of key switches currently available in 4x5 and 8x16 matrices. There are two types of keys switches, sealed for wet or dirty environments and keycap (in single, double or quad combinations).

It also provides up to 16 individually controllable LED indicators.

Typical products with the Model 301 and keypad module combination are the UMD Model 264 Custom Keyboard and the UMD Model 252 Wall Mount Keypad.

4.4 VGA Monitor Driver Module

The VGA monitor driver module provides standard VGA (640 x 480 pixel, monochrome and colour) output via its high density DB15 connector. This allows the family of *UMD ProtoLink Architecture* products to provide visual display unit functionality. Standard ANSI terminal emulation firmware is provided. This module also contains a bidirectional parallel port (ie either input or output).

A typical product with the Model 301 and VGA module combination is the UMD Model 291 Custom VGA Terminal.

5 UMD ProtoLink products currently manufactured

— Model 211 Custom LCD Terminal.
is made up of....
M301 CPU, 8 x 16 keyboard module (95 keys available), 2 x 40 operator display module
with optional expanded I/O module and magnetic card reader.

— Model 221 POS Peripheral Controller
is made of....
M301 CPU and expanded I/O module

— Model 250 Custom Industrial Aluminium Keyboard
is made up of....
M301 CPU and 8 x 16 keyboard module
with optional expanded I/O module.

— Model 264 Custom Keyboard
is made up of....
M301 CPU and 8 x 16 keyboard module
with optional expanded I/O module and magnetic card reader.

— Model 291 Custom VGA Terminal
is made up of....
M301 CPU and 8 x 16 keyboard module and VGA Driver module
with optional magnetic card reader.

— Model 330 Custom Wall mount Terminal
is made up of....
M301 CPU ,4 x 5 keyboard module and 2 x 16 operator display module
with optional expanded I/O.

— Model 331 Custom Wall mount Terminal
same as M330 but with Desk top case

— Model 363 Multi Keyboard/Serial Wedge
is made up of....
M301 CPU optional 2 x 16 operator display module.

— Model 390 Custom Panel mount Keyboard
is made up of....
M301 CPU and 8 x 16 keyboard module

— Model 392 Custom Panel Keyboard
is made up of....
M301 CPU and 4 x 5 keyboard module

— Model 490 Custom Industrial Aluminium LCD Terminal.
is made up of....
M301 CPU, 8 x 16 keyboard module (95 keys available),
2 x 40 operator display module
with optional expanded I/O module.

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

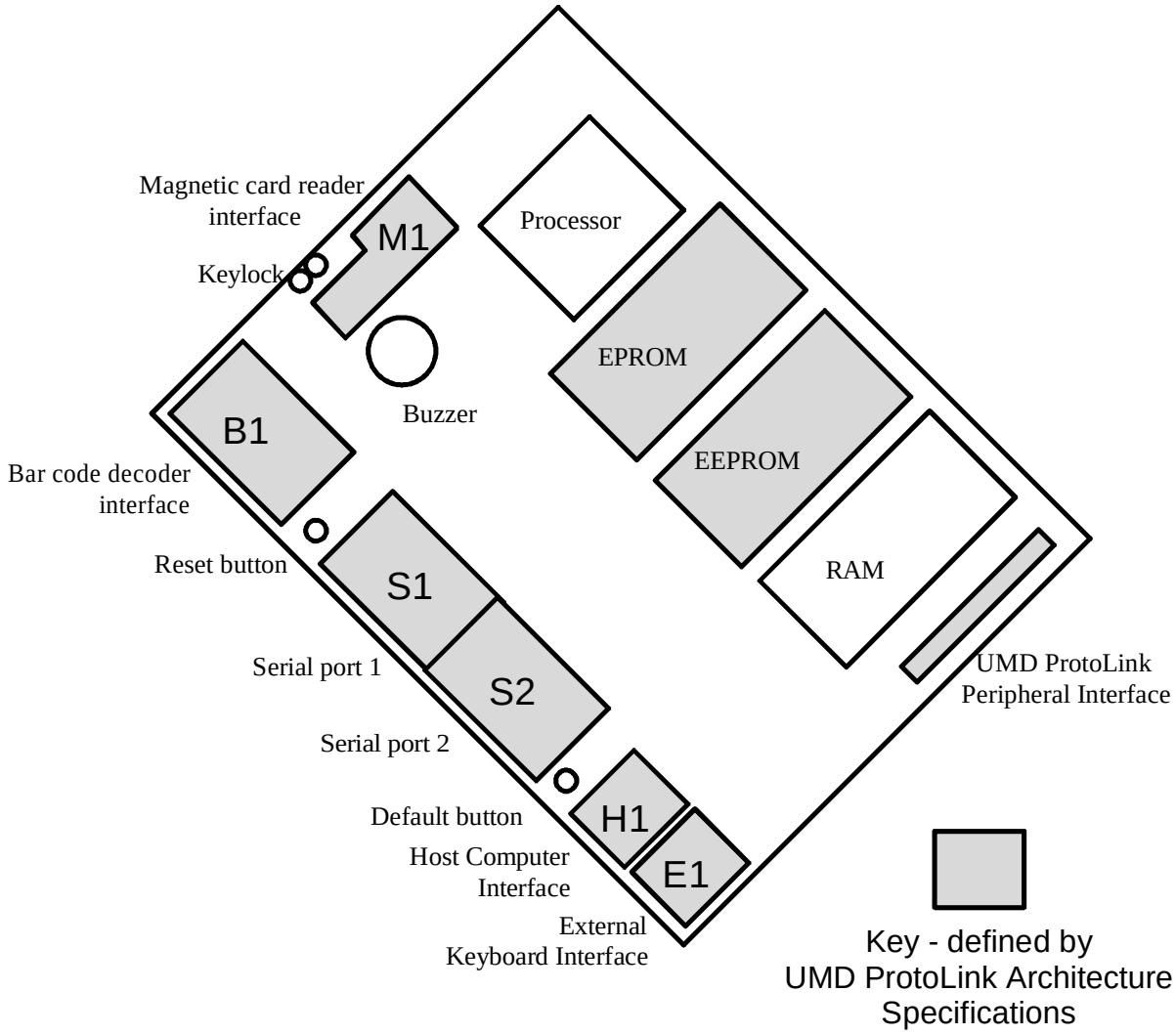
Chapter One
Hardware and Software Devices

1. Hardware Devices

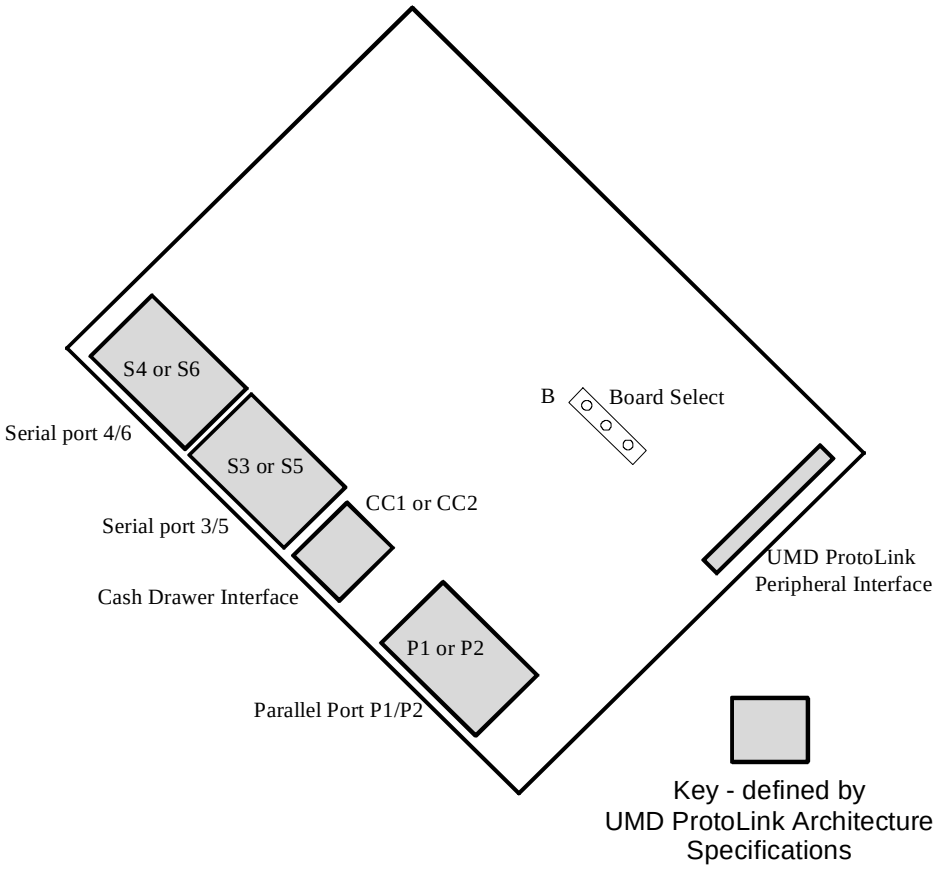
The UMD ProtoLink Architecture has defined a standard set of Hardware interfaces and devices. The following table shows the designated handles

Hardware Devices			
Device Handle		Description	Location
E1	input	External keyboard	M301 module
H1	I/O	Computer(host) keyboard	"
M1	input	Magnetic card reader	"
B1	input	Barcode reader	"
S1-S2	I/O	Serial ports	"
CB	out	Internal Buzzer	"
CC1-CC2	out	Cash drawer	I/O module
P1-P2	I/O	Parallel ports	"
S3-S6	I/O	Serial ports	"
CL	out	LEDS(1-16)	Keypad Matrix Module
K1	input	keypad matrix	"
D1-D2	out	LCD Displays	Operator Display Module
P3	I/O	Parallel port	VGA Monitor Driver
D3	out	VGA monitor driver	"

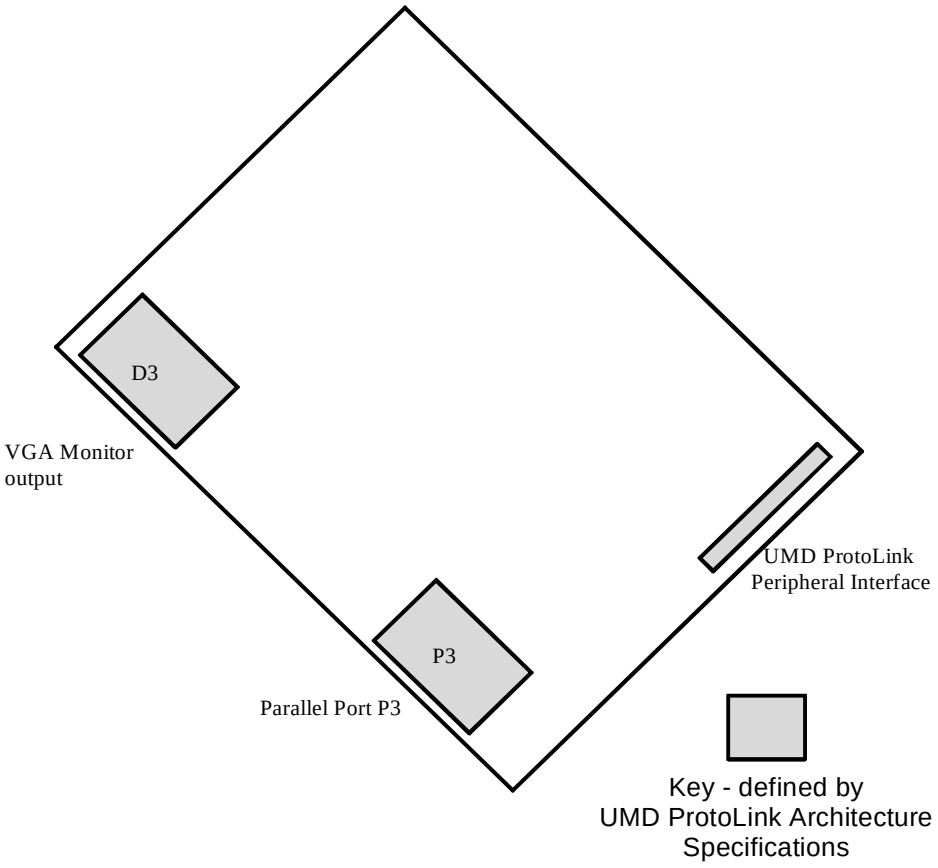
ProtoLink commands are accepted by the controller via its input ports where the incoming stream of data is checked for command sequences. These command sequences contain the "device handles" which tell the controller which interface or device the command is manipulating.



UMD Model 301Controller Module
Hardware Devices



UMD Model 301Expanded I/O Module
Hardware Devices



UMD Model 301VGA Monitor Driver Module
Hardware Devices

2. Software Devices

The UMD ProtoLink Architecture has defined a set of software specifications by which interfaces and devices are linked and controlled. One of these specifications is the ProtoLink Command Set, within this Command Set are the “Software Devices”. The following table shows the designated handles for these “Software Devices”.

Software Devices (filters and logical devices)		
Device Handle	Description	Device type
L0-L3	Logical I/O Device (L0 is standard)	Logical Device
V1-V8	Vector Device (used to split data streams into groups of 3 outputs)	Vector Device
x0	System Set, AT Key scan codes default	Filter Device
x1	ASCII to XT Key scan codes(PC-Term)	Filter Device
x2	ASCII to AT Key scan codes	Filter Device
x3	ASCII to IBM3197 Terminal Key scan codes	Filter Device
c0	Character filter (binary to hex)	Character Converter
c1	Uppercase conversion	Character Converter
c2	Lowercase conversion	Character Converter
c4	String conversion (instring - outstring)	String Converter
c5	Character to String conversion (charc1 -outstring1)	Character Converter
b1-b2	Block Devices	Block Device
d1-d3	Delay Device	Delay Device
i1-i3	Down Counter	Count Devive
N0-N2	Null Device,Current Device,Hold Device	Filter Device
T1	Touch Memory used through S2-S4 (Factory setup only)	Filter Device

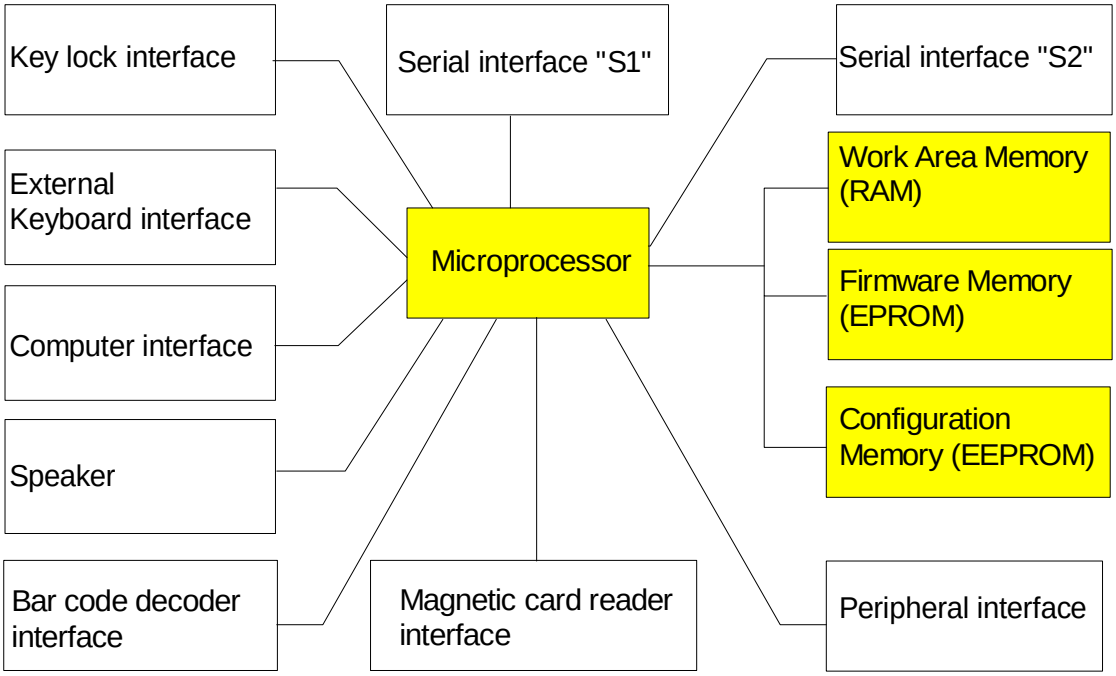
ProtoLink commands are accepted by the controller via its input ports where the incoming stream of data is checked for command sequences. Within the command sequences the software “device handles” are used to manipulate data streams, for character or string conversion, blocking devices, vectoring and filtering.

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Chapter Two
Model 301 Hardware Addresses

1. UMD Model 301 Controller Module

Unique Micro Design's premier family of *UMD ProtoLink Architecture* products is based on the **UMD Model 301 Controller Module** The Model 301 uses operating software (called firmware) to start the processor and do the house keeping tasks, similar to the BIOS in a PC. The firmware is located in EPROM.

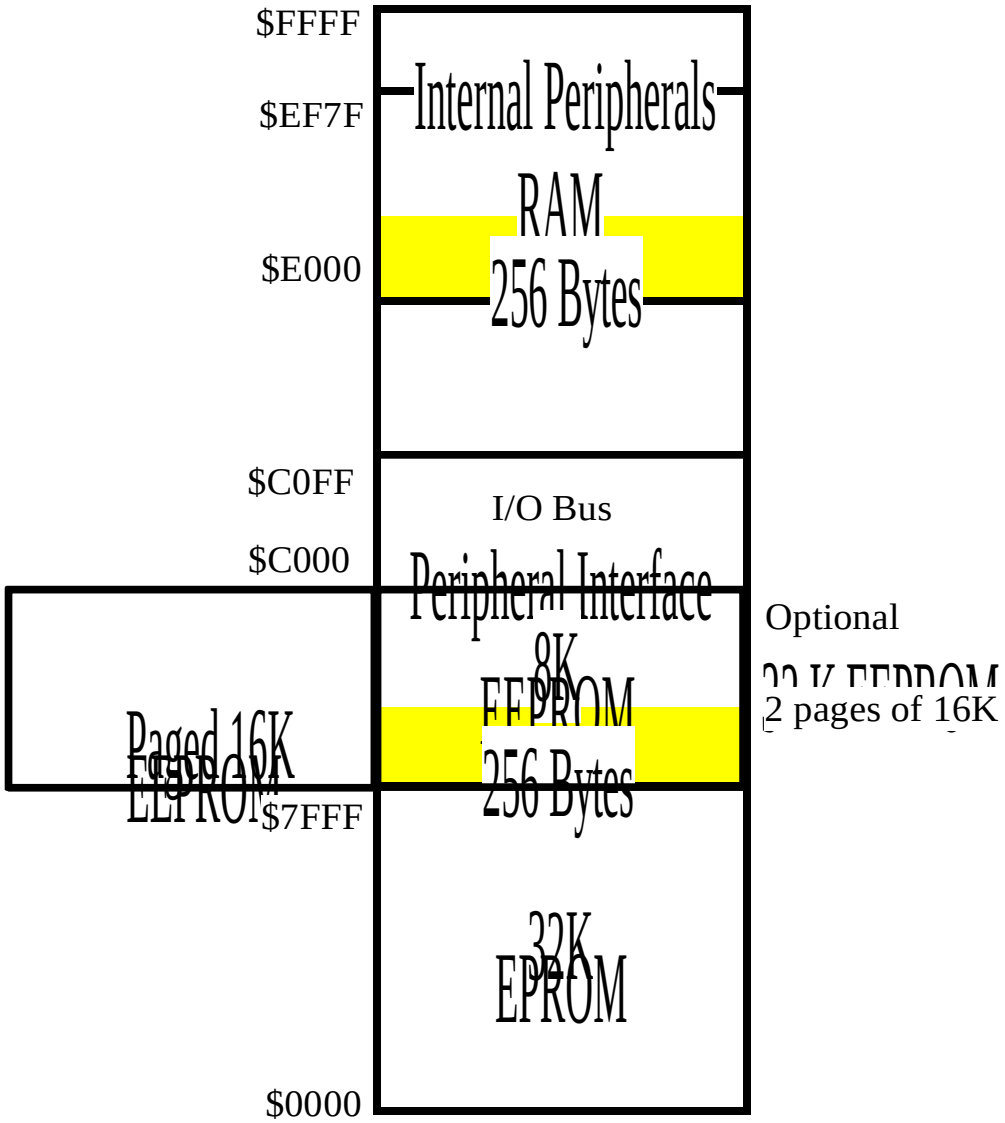


Model 301 Controller Module block diagram

The controller module also contains transitional work area memory, RAM, and a non volatile configuration memory area, EEPROM

These memory areas are located on the microprocessors "address space" and are located at the following address:

EPROM	$0000 - $7FFF
EEPROM	$8000 - $BFFF
Peripheral Interface	$C000 - $C0FF
RAM	$E000 - $FF7F
Internal Peripherals	$FF80 - $FFFF



Model 301 Controller Module Memory Addresses

2. Addressing the Model 301 memory

When the controller module is powered the microprocessor is reset and it looks at address “$0000” for its initialising vectors. Part of the initialisation program copies the first 256 bytes of the EEPROM to the RAM work area, to increase the operational speed of the system. After initialisation the firmware will always look at the values in RAM and not in the EEPROM, if any values are changed in this area of the EEPROM, the controller module must be reset to copy these changes to RAM.

The EEPROM is supplied as an 8K unit, if the operator writes to a location outside this 8K boundary the address is reflected back into the 8K address space, overwriting what was at that location. This is especially important when entering the values for the keys on the keyboard modules.

There is a 32K EEPROM option available, the microprocessor automatically switches from one 16K bank to the other when addressed.

The EEPROM is the configuration memory, the first 256 bytes have the values that the firmware uses for system set-ups and peripheral parameters. The remainder is used to store “User Strings” such as pre-ambles and post-ambles for the bar code decoder and the information that describes the values of the keys in the keyboard modules.

ProtoLink commands are accepted by the controller via its input ports where the incoming stream of data is checked for command sequences. Within the these command sequences the configuration memory is addressed using the command “EE” (EEPROM), this command writes data to the EEPROM and is discussed further in the chapter on Configuration Memory.

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Chapter Three
Configuration Memory

3

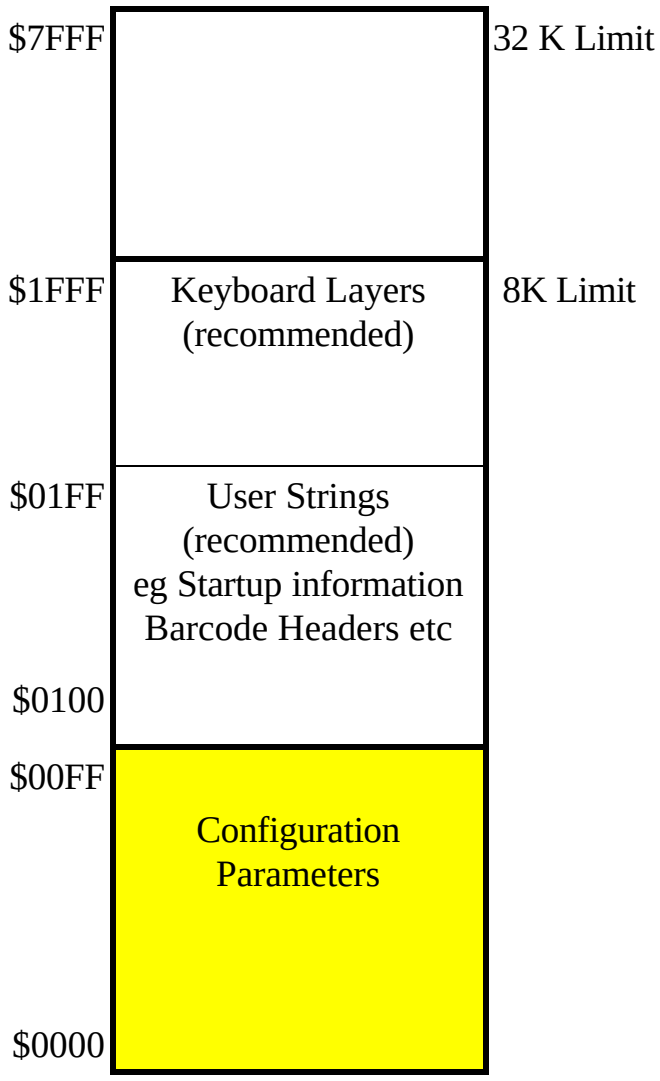
1. Configuration Memory

ProtoLink commands are accepted by the controller via its input ports where the incoming stream of data is checked for command sequences.

The addresses for the EE command start at $0000, this is interpreted by the contoller to mean: “talk to the EEPROM which is at address $8000 “.

The configuration memory address starts at $0000 and extends to $1FFF with the 8K EEPROM and to $7FFF with the 32K EEPROM.

The first 256 bytes ($0000 - $00FF) contain the system setup and peripheral parameters. The adresses $0100 - $01FF is recommended to be used for start up information, barcode headers and footers and blocking device information. The addresses $0200 - $1FFF is recommended to be used for the keyboard modules programmed key data.



Configuration Memory Adresses

2. Configuration Parameters

When the controller is initialised the first 256 bytes of the EEPROM are copied to the RAM work area, this is to increase the operational speed of the system. After initialisation the firmware will always look at the values in RAM and not in the EEPROM, if any values are changed in this area of the EEPROM, the controller module must be reset to copy these changes to RAM.

To change the contents of the EEPROM to a know set of values, that is Factory Default, the reset switch is drepressed while holding the default switch depressed (this is dicussed in detail in chapter later). The controller is reinitialised, the default values are copied into the EEPROM over writing its current contents. A reset is then generated and normal initialiation follows.

The addresses $0000 - $00FF (the first 256 bytes) of the EEPROM contain the configuration parameters. The configuration parameters are the attributes of the ProtoLink device, describing its personality. The values in each memory location, shown in the following tables, are the default values for the standard firmware (F00). The values for the F02 version of the firmware are also shown.

Address	Description	Default
00	Address of Startup String (msb).	0
01	Address of Startup String (lsb).	0
02	Buzzer Duration,Command Interpreter, LED Scan.	$F6
03	First byte used for the Command State Prefix.	2 [F02=$05 (^E)]
04	Second byte used for the Command State Prefix.	0 [F02=$19 (^Y)]
05	LED number for Num Lock.	12
06	LED number for Caps Lock.	13
07	LED number for Scroll Lock.	14
08	LED number for Good read from bar code or magnetic card.	15
09	LED number for Power.	16
0A	LED Flash Rate (20 ms increments).	25 (0.5sec)
0B	Type of keyboard used for the External Keyboard.	2 (AT) [F02=0]
0C	Type of computer used.	2 (AT)
0D	Delay for delay device one [d1] (1/100sec increments).	1 (10ms)
0E	Delay for delay device two [d2] (1/100sec increments).	2 (20ms)
0F	Delay for delay device three [d3] (1/100sec increments).	5 (50ms)
10	Address of Preamble for Block Device 1 [b1] (msb).	0
11	Address of Preamble for Block Device 1 [b1] (lsb).	0
12	Address of Postamble for Block Device 1 [b1] (msb).	0
13	Address of Postamble for Block Device 1 [b1] (lsb).	$32
14	Terminator character for Block Device 1 [b1].	$0D <CR>
15	Address of Preamble for Block Device 2 [b2] (msb).	0
16	Address of Preamble for Block Device 2 [b2] (lsb).	0
17	Address of Postamble for Block Device 2 [b2] (msb).	0
18	Address of Postamble for Block Device 2 [b2] (lsb).	$34
19	Terminator character for Block Device 2 [b2].	$0D <CR>
1A	Magnetic Card Track Data, Identifier and Multi Track info.	$07
1B	Address of Preamble for Magnetic Card Reader [M1] (msb).	0
1C	Address of Preamble for Magnetic Card Reader [M1] (lsb).	0
1D	Address of Postamble for Magnetic Card Reader [M1] (msb).	0
1E	Address of Postamble for Magnetic Card Reader [M1] (lsb).	$32

Default Values for Configuration Memory 00-1F

Address	Description	Default
1F	Address of Preamble for Bar Code Reader [B1] (msb).	0
20	Address of Preamble for Bar Code Reader [B1] (lsb).	0
21	Address of Postamble for Bar Code Reader [B1] (msb).	0
22	Address of Postamble for Bar Code Reader [B1] (lsb).	$32
23	Type of Bar Code to decode.	$1F
24	Various options for reading UPC type bar codes.	$CB
25	More options for reading UPC type bar codes.	$03
26	Various options for reading Codabar type bar codes.	$04
27	Transmit bar code symbology identifier and number of digits.	$00
28	Interleaved 2 of 5 first bar code length.	0
29	Interleaved 2 of 5 second bar code length.	0
2A		
2B	Reserved for Network Address.	1
2C	Speed, parity, data bits, stop bit values for serial port 1.	$42 (9600,N,8,1)
2D	Software and Hardware handshake options for serial port 1.	$0C (DTR/CTS) [F02=$07]
2E		
2F	Speed, parity, data bits, stop bit values for serial port 2.	$42 (9600,N,8,1)
30	Software and Hardware handshake options for serial port 2.	$0C (DTR/CTS) [F02=$07]
31		
32	Number of characters for Carriage Return String .	$01
33	Carriage Return ($0D).	$0D <CR>
34	Number of characters for Carriage Return/Line Feed String .	$02
35	Carriage Return ($0D).	$0D <CR>
36	Line Feed ($0A).	$0A <LF>
37		
38	Address of [c4] conversion (output str replaces input str) (msb).	0
39	Address of [c4] conversion (output str replaces input str) (lsb).	0
3A	Address of [c5] conversion table (output strings are replaced by input characters) (msb).	0
3B	Address of [c5] conversion table (output strings are replaced by input characters) (lsb).	0

Default Values for Configuration Memory 1F-3B

Address	Description	Default
3C	Address of Preamble for Touch Memory [T1] (msb).	0
3D	Address of Preamble for Touch Memory [T1] (lsb).	0
3E	Address of Postamble for Touch Memory [T1] (msb).	0
3F	Address of Postamble for Touch Memory [T1] (lsb).	$32
40	Down Character Counter 1[i1] - counts all characters.	1
41	Address of Down Counter 1 String. Outputs string when counter 1 reaches 0 (msb).	0
42	Address of Down Counter 1 String. Outputs string when counter 1 reaches 0 (lsb).	0
43	Down Character Counter 2 [i2] - counts all characters.	1
44	Address of Down Counter 2 String. Outputs string when counter 2 reaches 0 (msb).	0
45	Address of Down Counter 2 String. Outputs string when counter 2 reaches 0 (lsb).	0
46	Down Character Counter 3 [i3] - counts all characters.	1
47	Address of Down Counter 3 String. Outputs string when counter 3 reaches 0 (msb).	0
48	Address of Down Counter 3 String. Outputs string when counter 3 reaches 0 (lsb).	0
49		
4A		
4B		
4C		
4D		
4E		
4F		
50		
51		
52		
53		
54		
55		
.		
.		
7F		

Default Values for Configuration Memory 3C-7F

Address	Description	Default
80	Key click and key auto repeat rate.	$41
81	Rate A delay period (20 ms increments).	37 (740ms)
82	Rate A repeat rate (20 ms increments).	25 (500ms)
83	Rate B delay period (20 ms increments).	25 (500ms)
84	Rate B repeat rate (20 ms increments).	5 (100ms)
85		
86		
87	Base keyboard layer number.	128 [F02=129]
88	Address of first programmed keyboard layer (msb).	0 (none)
89	Address of first programmed keyboard layer (lsb).	0
8A		
8B		
8C	Parallel Port 1,2 and 3 direction.	$00
8D		
8E	Speed, parity, data bits, stop bit values for serial port 3.	$42 (9600,N,8,1)
8F	Software and Hardware handshake options for serial port 3.	$0C (DTR/CTS)
90		
91	Speed, parity, data bits, stop bit values for serial port 4.	$42 (9600,N,8,1)
92	Software and Hardware handshake options for serial port 4.	$0C (DTR/CTS)
93		
94	Speed, parity, data bits, stop bit values for serial port 5.	$42 (9600,N,8,1)
95	Software and Hardware handshake options for serial port 5.	$0C (DTR/CTS)
96		
97	Speed, parity, data bits, stop bit values for serial port 6.	$42 (9600,N,8,1)
98	Software and Hardware handshake options for serial port 6.	$0C (DTR/CTS)
99		
9A	Used Internally for VGA Driver Module	
9B		
9C		
9D		
9E		
9F		
.		
FF		

Default Values for Configuration Memory 80-FF

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Chapter Four
The Command Interpreter

4

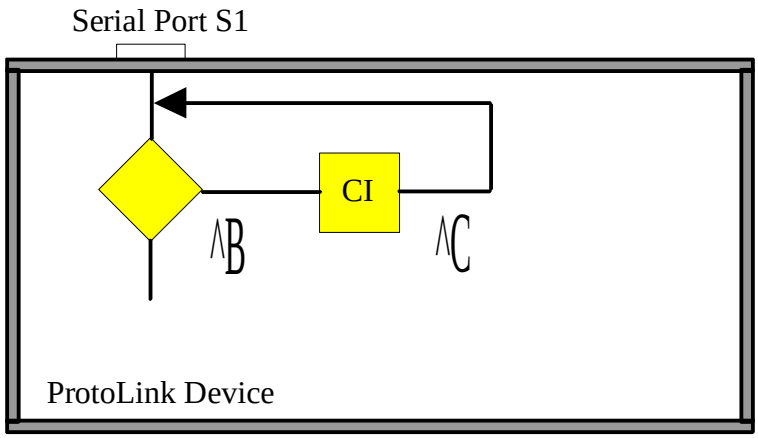
1. The Command Interpreter Explained

ProtoLink commands are accepted by the controller via its input ports where the incoming stream of data is checked for command sequences. To enable the command sequences to be acted upon the controller requires the “Command State Prefix” sequence, this then routes the stream of data to the Command Interpreter.

The Command State Prefix is determined by the value that is at the EE address $03, see Chapter 3, the standard firmware has the value $02 (^B).

The Command Interpreter interprets the “ProtoLink Command Set” of instructions which provide extensive control of interfaces and peripherals.

To exit the Command Interpreter the “Command State Suffix” is sent to the contoller. The Command State Suffix is a fixed character, $03 (^C).



The incoming stream of data into Serial port S1 being checked for the “Command State Prefix”

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Chapter Five
The ProtoLink Command Set

5

1. The ProtoLink Commands

The UMD ProtoLink Architecture has defined a set of software specifications by which interfaces and devices are linked and controlled. One of these specifications is the ProtoLink Command Set. The Command Interpreter interprets the ProtoLink Command Set executing the required operation as specified by the commands shown in the following tables :

Command		Created	Description
BE	i j k	29/7/92	Set bits indicated by bit mask **j** and clear bits with bit mask **k** for EEPROM at address **i**.
BI	i j k	29/7/92	Set bits indicated by bit mask **j** and clear bits with bit mask **k** for IOBUS at address **i**.
BR	i j k	29/7/92	Set bits indicated by bit mask **j** and clear bits with bit mask **k** for RAM/ROM at address **i**.
CB	i j	29/7/92	Contol Buzzer by bit mask **i** and action code **j**. Action code **0** Off, **1** On, **2** Toggle (25/8/94), **3** Trigger (25/8/94), **4** Flash (25/8/94). eg CB 1 2
CC	i j	29/7/92	Control Cash Drawer by bit mask **i** and action code **j**. Action code **0** Off, **1** On. [CC 1]
CL	i j	29/7/92	Control LED(s) by bit mask **i** and action code **j**. Action code **0** Off, **1** On, **2** Toggle, **3** Trigger, **4** Flash. [eg CL $0200 4]
DE	i [j]	29/7/92	Dump **j** bytes from EEPROM memory starting from address **i**. [eg DE $0100 50]
DG	1..3	29/7/92	Dump Global variable **i**. **1 =** IO linkages, **2** = Firmware version, **3** = Firmware module revisions. [eg DG 1]
DI	i [j]	29/7/92	Dump **j** bytes from IOBUS memory starting from address **i**.
DL	i	29/7/92	Delay for **i**/100 seconds. (range of **i**: 0 to $3FFF) [eg DL 100 - 1 second].
DR	i [j]	29/7/92	Dump **j** bytes from RAM/ROM memory starting from address **i**.
EE	i j..	29/7/92	Enter bytes **j..** to EEPROM starting from address **i**. [eg EE $0000 $01 $00]
EI	i j..	29/7/92	Enter bytes **j..** to IOBUS starting from address **i**.
ER	i j..	29/7/92	Enter bytes **j..** to RAM/ROM starting from address **i**.

ProtoLink Commands BE - ER

Command	Created	Description
KB i	29/7/92	Make keyboard layout **i** the Base layout temporarily (EE location $87 is base layout on power up).
KD	29/7/92	Deselect current keyboard layout.
KH i	29/7/92	Hold keyboard layout **i**. Temporarily locks layout **i** until after a second key is pressed.
KL i	29/7/92	Lock keyboard layout **i**. Makes the specified layout the locked layer, KU unlocks.
KM i j	29/7/92	Key Mode. Where **i** = mode; **0** never pressed, **1** no auto repeat, **2** auto repeat rate A, **3** auto repeat rate B. **j** = key number (**0** = all keys).
KP i	29/7/92	Press (ie make) key number **i**.
KR i	29/7/92	Release (ie break) key number **i**.
KS i	29/7/92	Select keyboard layout **i** (KD deselects layer).
KU	29/7/92	Unlock current keyboard layout.
LI id od	29/7/92	Link input device **id** to output device **od**. [eg LI L0 S1]
OE i [j]	29/7/92	Output bytes from EEPROM starting from address **i**. If **j** = 0 or not provided then the bytes are output until a zero (null) char is found. If **j** is entered then j bytes are output.
OI i [j]	29/7/92	Output bytes from IOBUS starting from address **i**. If **j** = 0 or not provided then the bytes are output until a zero (null) char is found. If **j** is entered then j bytes are output.
OR i [j]	29/7/92	Output bytes from RAM/ROM starting from address **i**. If **j** = 0 or not provided then the bytes are output until a zero (null) char is found. If **j** is entered then j bytes are output.
PD od i	29/7/92	Put bytes **i..** to output device **od**. [eg PD D1 "Display"]
RD	19/8/94	Reset back to defaults.
RE	29/7/92	Issue immediate software reset.

ProtoLink Commands KB - RE

2. Optional Real Time Clock Commands

The UMD ProtoLink Architecture has an optional Real Time clock. The commands below should only be used when this is installed.

Attempting to use the time and date commands when the timer is NOT installed will result in damage to the EEPROM

Command	Created	Description
GT	29/7/92	Outputs the time in the form, hh:mm:ss
GD	29/7/92/	Outputs the date in the form, dd/mm/yy
ST hhmmss	29/7/92	Sets the real time clock
SD ddmmyy	29/7/92	Sets the real time clock

ProtoLink Commands GT - SD

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Chapter Six
Defining
the Configuration Parameters

6

1 The Start Up String

When the controller module is powered the microprocessor is reset and begins its initialisation. Part of the initialisation sends information to the Command Interpreter which creates a table in RAM defining the state of the software and hardware devices

The table that is created defines ... " what device is *linked* to what device " for example the standard firmware (F00) links serial port 1 (S1) to the standard I/O device (L0).

This "Linkage" is established when the command sequence;

LI S1 L0

is sent to the command interpreter.

Link commands are futher explained in Chapter 7 and the list of default linkages is found in the Appendix.

After the table is created, the initialisation process then checks the configuration memory, at $0000, for the location of a user " Start Up String" (also called autoexec).

The Start Up String is used to set up linkages that differ from the default, it can also be used to issue other commands to the command interpreter upon start up, such as turning on indicator LED's or sending information to serial ports.

2 Variable length strings, Pointers and Bit Flags

The configuration memory contains the parameters that are the attributes of the ProtoLink device, describing its personality. The values in each memory location can be one of three types: variable length strings, pointers or bit flags.

Variable length strings are a sequence of characters, preceded by a length attribute byte which indicates the number of characters in the string.

For example at configuration memory address $0034-$0036, the “carriage return/ line feed string” is specified. Address $0034 contains the string length .. $02... Address $0035 and $0036 contain the string characters.

Pointers are used in the configuration memory to direct the controller to another configuration memory address, which is the location of the data that a specific command requires.

For example at configuration memory address $0000 and $0001are the values that represent the configuration memory address of the data that makes up the start up string. $0000 contains $01 and $0001 contains $00 which tells the controller the data is located at $0100.

Each “byte” of memory contains contains “8 bits”, each bit can be the binary value of either 0 or 1. The bits are number 0 (lsb) to 7 (msb)

Bit flag type configuration parameters are either specified as individual bits which can signify a specific option being enabled (binary 1) or disabled (binary 0), or they are specified as a number of consecutive bits with the bit pattern specifying a particular option.

Bit number								
most significant bit	7	6	5	4	3	2	1	0	least significant bit

Bit order within a byte

7	6	5	4	3	2	1	0	Bit number
Buzzer duration								
						0	0	none
						0	1	short
						1	0	medium
						1	1	long

Example of a group of bits specifying options

7	6	5	4	3	2	1	0	Bit number
Command interpreter								
					1			enabled on device S1 (0 disabled)

Example of an individual bit flag

3 Configuration Parameter Specifications

This section describes in detail the attributes of each of the configuration parameters, and how this may effect the personality of the UMD ProtoLink device.

3.1 Start Up String

Points to the address of the start up string.

Address	Description	Default
$00	Address of startup string (msb) User $0000 if no startup string	0
$01	Address of startup string (lsb)	0

3.2 Operating Modes

3.2.1 Buzzer and Command Interpreter

Sets modes for the buzzer, and the ports that can be used to talk to the command interpreter.

On reset the controller enables each LED in turn, this can be used to visually check the LED's. This option can be disabled.

Address								
$02								
7	6	5	4	3	2	1	0	Bit number
Buzzer duration								
						0	0	none
						0	1	short
						1	0	medium
						1	1	long
Command interpreter								
					1			enabled on device S1 (0 disabled)
				0				disabled on device S2 (1 enabled)
			1					enabled on device P1 (0 disabled)
		0						disabled on device H1 (1 enabled)
	0							disabled on device B1 (1 enabled)
LED Scan								
0								enabled (1 to disable)
Default value								
0	0	0	1	0	1	1	0	$16

3.2.2 Command State Prefix

If necessary the command state prefix can be changed. This may be required in a terminal serial connection where control codes are used for terminal manipulation. Either a single or dual byte sequence can be defined.

Address	Command State Prefix for devices S1, S2 and P1	Default
$03	First prefix	2 ie <STX>
$04	Second prefix (use 0 if none)	0

3.2.3 Indicator LED Options

For products which provide keyboard functionality, LED indicators can be assigned for “Num Lock”, “Caps Lock” and “Scroll Lock”. Indicators can also be assigned for “Power on” and a good read indicator for the magnetic card reader and the BCR port.

LED indicators can also be programmed to flash, the flash rate can also be set in 20 mS intervals.

Address	Indicator number assignments	Default
$05	Num lock, 0 = none	12
$06	Caps lock, 0 = none	13
$07	Scroll lock, 0 = none	14
$08	Good read, 0 = none	15
$09	Power on, 0 = none	16

Address	Description	Default
$0A	LED flash rate (20 mS increments)	25 ie 0.5 S

3.3 Computer and External keyboard Interface

The H1 and K1 interface ports can be selected as one of a number of different type including having no connection.

The x0 software device filters data according to the Host computer keyboard type set at address $0C.

Address $0B	
Value	**External keyboard type**
0	No external keyboard attached
1	XT
2	AT
3	AT interface but XT scan codes
4	Reserved
2	Default value

Address $0C	
Value	**Computer interface**
0	No computer attached
1	XT
2	AT
3	AT interface but XT scan codes
4	Reserved
2	Default value

3.4 Intercharacter Delay Device

These software devices are used to delay the dataflow from device to device, for each of the three devices a different default delay is set.

Address	Intercharacter delay in 1/100 sec increments	Default
$0D	Delay device one (d1)	1 ie 10 mS
$0E	Delay device two (d2)	2 ie 20 mS
$0F	Delay device three (d3)	5 ie 50 mS

3.5 Block Devices

The “Block Devices” provide a function to place a preamble and a postamble string around a block of data. There are two block devices available, “b1” and “b2”.

Input into a block device is buffered until a terminating character is recieved. Once this character is received, the preamble string is output followed by the buffered characters (excluding the terminating character), followed by the postamble string.

The terminating character is specified in the configuration parameters, specified at address $14 for b1 and $19 for b2. The buffer for “b1” is 100 bytes and for “b2” is 50 bytes. Both the preamble and postamble string can be a maximum of 255 bytes.

Data (Max 100 (50) bytes)	CR (terminating character)

If the incomming data exceeds the buffer size before receiving the terminating charcter, the buffer is output. This output is the preamble string followed by the buffered data but not the postamble string. The buffer will fill again while waiting for the terminating character.

Preamble Max 255 bytes	Data	Postamble Max 255 bytes

The block devices b1 (b2) add preamble and postamble strings to a block of data.

The address of the preamble and postamble string is placed in memory at the locations shown below. The controller obtains the pre/postamble string from these locations, the first byte indicates how many bytes of data in the string, see Chapter 7 for further explanation.

Address	Device One (b1) Pre/Postamble strings	Default
$10	Preamble (msb)	0
$11	Preamble (lsb)	0
$12	Postamble (msb)	0
$13	Postamble (lsb)	$32
$14	Block 1 terminator	$0D ie <CR>

Address	Block Device Two (b2) Pre/Postamble strings	Default
$15	Preamble (msb)	0
$16	Preamble (lsb)	0
$17	Postamble (msb)	0
$18	Postamble (lsb)	$34
$19	Block 2 terminator	$0D ie <CR>

3.6 Magnetic Card Reader

3.6.1 Track Data and Identifiers

Magnetic card readers are available for tracks 1, 2, or 3. Each track can be enabled individually or combination (depending on the magnetic card reader hardware).

The **track identifier**, when enabled, is an ASCII number (1,2,3) sent as the first byte before the data for that track. The example show has all the possible data fields.

Pre amble	1	Track 1 Data	Post amble	Pre amble	2	Track 2 Data	Post amble	Pre amble	3	Track 3 Data	Post amble

The **order** in which the track data is sent, can be selected. The data can also be sent **concatenated or not concatenated**. Only the data fields that are output in default mode are shown here.

Concatenated default order

Track 1 Data	Track 2 Data	Track 3 Data	CR

Not concatenated reverse order

Track 3 Data	CR	Track 2 Data	CR	Track 1 Data	CR

Empty Track Option disabled : ANY enabled track which has valid data, has that data sent and is recognised as a good read (indicator flashed if enabled), empty tracks are not sent.
Example : If all three tracks are selected, but there is only one track of data on the card (track 2) only the data from this track is sent, note the 'CR' is the default postamble character.

	Track 2 Data	CR

Empty Track Option enabled : ALL enabled tracks which have valid data, have that data sent and recognised as a good read, empty tracks are sent with the specified postamble and preamble. If any track has bad data, then no data is sent and is recognised as a bad read (no good read indication).

Example : If all three tracks are selected, but there is only one track of data on the card (track 2), an empty field plus pre/post amble for track 1, the data plus pre/post amble for track 2 and an empty field plus pre/post amble for track 3 are sent, note the 'CR' is the default postamble character.

	CR	Track 2 Data	CR		CR

Address $1A								
7	6	5	4	3	2	1	0	Bit number
								Send Magnetic Card Track Data
							1	Track 1 (0 to disable)
						1		Track 2
					1			Track 3
								Track Order Options
				0				send tracks t1,t2,t3 (1 sets order t3,t2,t1)
			0					track data not concatenated (1 to enable)
								Reserved
		0						
								Track Identifier
	0							not sent (1 to send)
								Empty Track Option
0								disabled (1 to enable) see previous page
								Default value
0	0	0	0	0	1	1	1	$07

3.6.2 Magnetic Card Preamble and Postamble

The address of the preamble and postamble string is placed in memory as shown below. The controller obtains the pre/postamble string from these locations, the first byte indicates how many bytes of data in the string, see Chapter 7 for further explanation.

Address	Magnetic Card Preamble/Postamble strings	Default
$1B	Preamble (msb)	0
$1C	Preamble (lsb)	0
$1D	Postamble (msb)	0
$1E	Postamble (lsb)	$32

3.7 Bar Code Decoder Interface Options.

The bar code decoder interface (B1) has options for postambles and preambles and symbology options for ITF, Code 39, Code 128, CODABAR and APN/EAN/UPC. There are also three type of bar code symbology identifier available.

3.7.1 Preambles and Postambles.

The address of the preamble and postamble string is placed in memory as shown below. The controller obtains the pre/postamble string from these locations, the first byte indicates how many bytes of data in the string, see Chapter 7 for further explanation.

Address	Bar Code Decoder Pre/Postamble strings	Default
$1F	Preamble (msb)	0
$20	Preamble (lsb)	0
$21	Postamble (msb)	0
$22	Postamble (lsb)	$32

3.7.2 Bar Code Symbologies

Address $23								
7	6	5	4	3	2	1	0	Bit number
								Decode Interleaved 2 of 5
							1	enabled (0 to disable)
								Decode Code 39
						1		enabled (0 to disable)
								Decode Code 128
					1			enabled (0 to disable)
								Decode Codabar
				1				enabled (0 to disable)
								Decode APN/EAN/UPC
			1					enabled (0 to disable)
								Reserved
0	0	0						
								Default value
0	0	0	1	1	1	1	1	$1F

3.7.3 Bar Code Symbologies Options

Address $24								
7	6	5	4	3	2	1	0	Bit number
								UPC A - Transmit check digit
							1	Enabled (0 disabled)
								UPC A - Transmit number system digit
						1		Enabled (0 disabled)
								UPC E - Transmit check digit
					0			disabled (1 Enabled)
								UPC E - Transmit number system digit
				1				Enabled (0 disabled)
								UPC E - Expand UPC E to UPC A
			0					disabled (1 Enabled)
								EAN/UPC Addons
		0						*not implemented*
								EAN/UPC Addons - 2 digit
	1							*not implemented*
								EAN/UPC Addons - 5 digit
1								*not implemented*
								Default value
1	1	0	0	1	0	1	1	$CB

Address $25								
7	6	5	4	3	2	1	0	Bit number
								Expand UPC-A to EAN 13
							1	Enabled (0 to disable)
								Extended Code 39
						1		Enabled (0 to disable)
								Pad EAN 8 to EAN 13
					0			disabled (1 to Enable)
								Reserved
0	0	0	0	0				
								Default value
0	0	0	0	0	0	1	1	$03

3.7.4 Codabar Symbology Options

Address $26								
7	6	5	4	3	2	1	0	Bit number
								Codabar - verify check digit
							0	disabled (1 to Enable)
								Codabar - transmit check digit
						0		disabled (1 to Enable)
								Codabar - transmit start/stop character
					1			Enabled (0 to disable)
								Codabar - concatenation
				0				disabled (1 to Enable)
								Reserved
0	0	0	0					
								Default value
0	0	0	0	0	1	0	0	$04

3.7.5 Bar Code Symbology Identifies / Number of Digits

The bar code symbology identifiers identify the type of symbology read by the bar code decoder. This identifier can be sent as a preamble to the data, after the preamble string. The number of digits (characters) that is contained in the data can also be calculated and added as a preamble, this does not apply to the EAN/UPC symbologies.

For example, when the “Type 1” identifier is enabled and the number of digits has been enabled, reading a Code 39 bar code ... “12345” the ASCII output would be:

Preamble	Identifier	No. digits	Data	CR

“M0512345”

Symbology	Type 1	Type 2	Type 3
UPC-A	“A”	“A”	“c”
UPC-E	“C”	“E”	“c”
EAN-8	“B”	“FF”	“d”
EAN-13	“A”	“F”	“d”
Code 39	“M”	“*”	“b”
Codabar	“N”	“%”	“a”
Code 93	“L”	“&”	“i”
Code 128	“K”	“#”	“j”
ITF	“I”	“i”	“e”

Bar Code Symbology Identifiers

Address $27								
7	6	5	4	3	2	1	0	Bit number
								Transmit bar code symbology identifier, see text for explanation
						0	0	disabled
						0	1	Type 1
						1	0	Type 2
						1	1	Type 3
								Transmit Number of digits, excludes EAN / UPC symbologies
					0			disabled (1 Enable)
								Reserved
0	0	0	0	0				
								Default value
0	0	0	0	0	0	0	0	$00

3.7.6 Interleaved 2 of 5 Options

Address	ITF Length options	Default
$28	Length option 1 (0 equals variable length data) The selected value can be the range 1- $7E (126)	0
$29	Length option 2 (0 equals no second length) The selected value can be the range 1- $7E (126)	0

3.8 Network Address

The Network Address provides a unique number for the ProtoLink Device in the range $0-$FF (0-255).

Address	Description	Default
$2B	Network Address	1

3.9 Serial Communication Port Parameters

3.9.1 Communication Port Definitions

The communication parameters for these ports are programmed in two groups. The first group “Format” contains the settings for baud rate, Stop Parity and Data bits. The second group “Handshake” contains the options for handshaking which are XON / XOFF transmit and receive, DTR and CTS handshaking.

The bit flag configuration parameters are defined in the following tables.

7	6	5	4	3	2	1	0	Bit number
								Baud Rate
					0	0	0	34400
					0	0	1	19200
					0	1	0	9600
					0	1	1	4800
					1	0	0	2400
					1	0	1	1200
					1	1	0	600
					1	1	1	300
								Stop bits
				0				1
				1				2
								Parity type
			0					Even
			1					Odd
								Parity
		1						...Enabled (0 to disable)
								Data bits
	0							7
	1							8
								Reserved
0								

Serial Port Format Definition Table

7	6	5	4	3	2	1	0	Bit number
								XON/XOFF receive handshake
							1	Enabled (0 to disable)
								XON/XOFF transmit handshake
						1		Enabled (0 to disable)
								DTR handshake
					1			Enabled (0 to disable)
								CTS handshake
				1				Enabled (0 to disable)
								Reserved
0	0	0	0					

Serial Port Handshake Definition Table

3.9.2 Serial Port S1 and S2

The addresses of the communication parameters for S1 and S2 are shown below, indicating the definition table that is used to manipulate the bit flags.

Address	Serial port one (S1) parameters	Default
$2C	Format definition table	$42 (ie 9600,N,8,1)
$2D	Handshake definition table	$0C (ie DTR/CTS h/s)
$2E	Reserved	0

Address	Serial port two (S2) parameters	Default
$2F	Format definition table	$42 (ie 9600,N,8,1)
$30	Handshake definition table	$0C (ie DTR/CTS h/s)
$31	Reserved	0

3.10 Default Strings

The default strings are “variable length strings” as described in section 2 of this chapter. They are used by a number of programmable functions, including the default postamble for magnetic card reader and bar code decoder. It is advisable not to change these values.

Address	Default Strings	Default
$32	Carriage Return (<CR>)	$01 $0D
$34	Carriage Return/Line feed (<CR><LF>)	$02 $0D $0A

3.11 String Conversion

The software devices “c4” and “c5” find the addresses of the conversion data at $38 (msb), $39 (lsb) and $3A (msb), $3B (lsb) respectively.

The “c4” device requires the length and value of the “input string” and the length and value of the “output string”

The “c5” device can have up to 255 possible characters to convert, it requires the “input character”, the length of the “output string” and the number of bytes in the output string.

Examples of these conversions are explained in detail in chapter 7.

Address	“c4” and “c5” String Converters	Default
$38	Address of C4 data (msb)	0
$39	Address of C4 data (lsb)	0
$3A	Address of C5 table (msb)	0
$3B	Address of C5 table (lsb)	0

3.12 Touch Memory Pre/Postamble

The Touch Memory software device has programmable preamble and postamble, the address of the preamble and postamble string is placed in memory as shown below. The controller obtains the pre/postamble string from these locations, the first byte indicates how many bytes of data in the string, see Chapter 8 for further explanation.

Address	Touch Memory Pre/Postamble strings	Default
$3C	Preamble (msb)	0
$3D	Preamble (lsb)	0
$3E	Postamble (msb)	0
$3F	Postamble (lsb)	$32

3.13 Character Counters

The software devices “i1”, “i2”, and “i3” are used to count the number of characters and append a string. The count number is located at $40 for “i1”, $43 for “i2” and $46 for the “i3”. The addresses of the associated output string is shown in the table below. The controller obtains the counter output string from these locations, the first byte indicates how many bytes of data in the string, see Chapter 7 for further explanation.

Address	Character Counters “i1”, “i2” and “i3”	Default
$40	Down Character Counter 1 ..count	1
$41	Address of Output String “i1” (msb)	0
$42	Address of Output String “i1” (lsb)	0
$43	Down Character Counter 2 ..count	1
$44	Address of Output String “i2” (msb)	0
$45	Address of Output String “i2” (lsb)	0
$46	Down Character Counter 3 ..count	1
$47	Address of Output String “i3” (msb)	0
$48	Address of Output String “i3” (lsb)	0

3.14 Keyboard Matrix Controls

The hardware device “K1”, the keyboard matrix, can have each of the keys programmed to output a series of characters. The controller locates the information by reading the values about the keyboard matrix at addresses $87 - $89. Other keyboard matrix controls are “key click” sound, “auto repeat” and “auto repeat rate”

3.14.1 Key Click, Auto key repeat

Address								
$80								
7	6	5	4	3	2	1	0	Bit number
Key click								
							1	Enabled
Reserved								
		0	0	0	0	0		
Key auto repeat rate								
0	0							key disabled
0	1							no repeat
1	0							repeat with rate A
1	1							repeat with rate B
Default value								
0	1	0	0	0	0	0	1	$41

3.14.2 Key Repeat Rates

Address	Key repeat rates in 20mS increments	Default
$81	Rate A delay period (valid range 1 to $37 (55))	$25(37)
$82	Rate A repeat rate (valid range 1 to $37 (55))	$19(25)
$83	Rate B delay period (valid range 1 to $37 (55))	$19(25)
$84	Rate B repeat rate (valid range 1 to $37 (55))	$05(5)

3.14.3 Base Keyboard layer and Address Programmed keyboard Layer

Address	Description	Default
$87	Base keyboard layer number	$80 (128)
$88	Address of first programmed keyboard layer (msb) $0000 if none	0
$89	" " " " " (lsb)	0

3.15 Parallel Ports "P1", "P2" and "P3"

The parallel ports are defined as either an input or an output.

Address $8C								
7	6	5	4	3	2	1	0	Bit number
Parallel port one (P1) direction								
							0	Input
							1	Output
Parallel port two (P2) direction								
						0		Input
						1		Output
Parallel port three (P3) direction								
					0			Input
					1			Output
Reserved								
0	0	0	0	0				
Default value								
0	0	0	0	0	1	1	1	$07

3.16 Serial Port S3, S4, S5 and S6

The addresses of the communication parameters for S3, S4, S5 and S6 are shown below, indicating the definition table that is used to manipulate the bit flags.

See Section 3.9.1 for the definition tables and there explanation.

Address	Serial port one (S3) parameters	Default
$8E	Format definition table	$42 (ie 9600,N,8,1)
$8F	Handshake definition table	$0C (ie DTR/CTS h/s)
$90	Reserved	0

Address	Serial port one (S4) parameters	Default
$91	Format definition table	$42 (ie 9600,N,8,1)
$92	Handshake definition table	$0C (ie DTR/CTS h/s)
$93	Reserved	0

Address	Serial port one (S5) parameters	Default
$94	Format definition table	$42 (ie 9600,N,8,1)
$95	Handshake definition table	$0C (ie DTR/CTS h/s)
$96	Reserved	0

Address	Serial port one (S6) parameters	Default
$97	Format definition table	$42 (ie 9600,N,8,1)
$98	Handshake definition table	$0C (ie DTR/CTS h/s)
$99	Reserved	0

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Chapter Seven
Programming and Communicating with the ProtoLink Device

7

1 What can be achieved by programming the ProtoLink Product ?

In the previous chapters we have discussed the ProtoLink Command Set, configuration parameters, hardware and software devices and the command interpreter. When all this information is put into practice, we can communicate with and control the powerful functions built into the ProtoLink Architecture.

When programming or communicating with the ProtoLink device, temporary and permanent changes (until factory defaulted) can be made to the default personality.

—— Permanent Changes: programming the controller with a start up string. This was discussed in Chapter 6.

—— Permanent Changes: programming the controller with the values for the keypad matrix.

—— Permanent Changes: writing to the Configuration Memory in the EEPROM. These instructions are sent to the controller using the EE command, for example, Changing the communication parameters for S2.

—— Temporary Changes:
These are instructions that are sent to the controller that remain active until changed or the power is removed, example :
Linking S1 port to S2 port

2 Connecting the ProtoLink Product

ProtoLink products can be used in keyboard mode or serial mode

Keyboard mode
- compatible with PC/XT/AT and PS/2 keyboards.
- standard keyboard can be used in series. (keyboard wedging)
- operation is transparent to computer.

Serial mode
- one serial port for host communication
- second serial port for attached serial input.
- can be used as a serial wedge between terminal and host

2.1 Connecting the ProtoLink Product to a computer keyboard port "Keyboard mode"

1) Turn off the PC.

2) Disconnect standard PC Keyboard.

3) Referring to figure 1, connect one end of the UMD CA202 or CA226 cable to the PC's keyboard interface connector.

4) Connect the other end of the CA202 or CA226 cable to the 7 pin DIN "Comp Ext Pwr" (Computer and External Power) connector on the UMD ProtoLink Product. Note that the 5 pin DIN connector on the CA202 or CA226 cable mates with 7 pin DIN connector.

5) Attach the standard PC Keyboard to the 5 pin DIN "Ext Kbd" (External Keyboard) connector on the UMD ProtoLink Product. If the keyboard uses a six pin mini DIN connector, you will have to use a CA227 adapter to convert the PC Keyboard's connector to that on the UMD ProtoLink Product.

6) Turn on the PC.

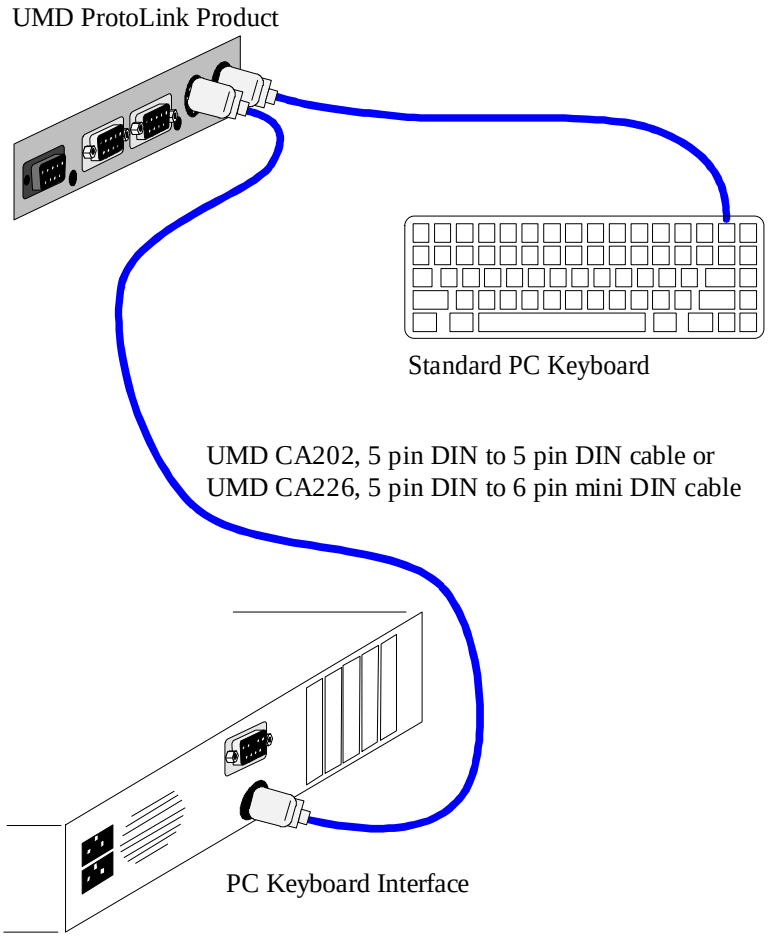


Figure 1

2.2 Connecting a computer serial port (or terminal) to the ProtoLink Product. "Serial Mode"

1) Turn off the PC.

2) Referring to figure 2, connect the UMD CA201 or CA211 serial cable to either the S1 or S2 serial port on the UMD ProtoLink Product.

2) Connect the other end of the serial cable to the COM1: or COM2: serial port on the PC.

3) Connect the power pack to the 7 pin DIN "Comp Ext Pwr" (Computer and External Power) connector.

4) Connect the power pack to the mains power and turn it on.

5) Turn on the PC.

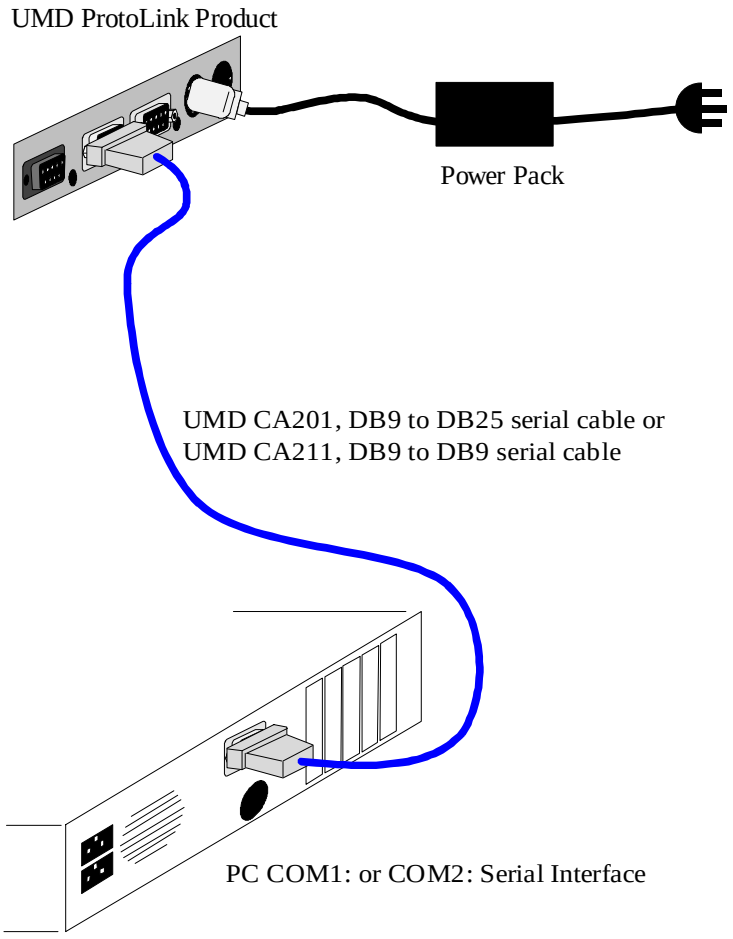


Figure 2

2.3 Connecting the ProtoLink Product between a terminal and host computer. "Serial Mode"

1) Referring to figure 3, unplug the cable on the terminal (from the host) connect the UMD CA6027 cable to the S2 serial port on the UMD ProtoLink Product.

2) Connect the other end of the cable to the modem or input port on the terminal (the port the host was connected to).

3) Connect the UMD CA6028 cable to the S1 serial port on the UMD ProtoLink Product.

4) Connect the other end of the cable, to the original cable, that went to the terminal from the host.

5) Connect the power pack to the 7 pin DIN "Comp Ext Pwr" (Computer and External Power) connector.

6) Connect the power pack to the mains power and turn it on.

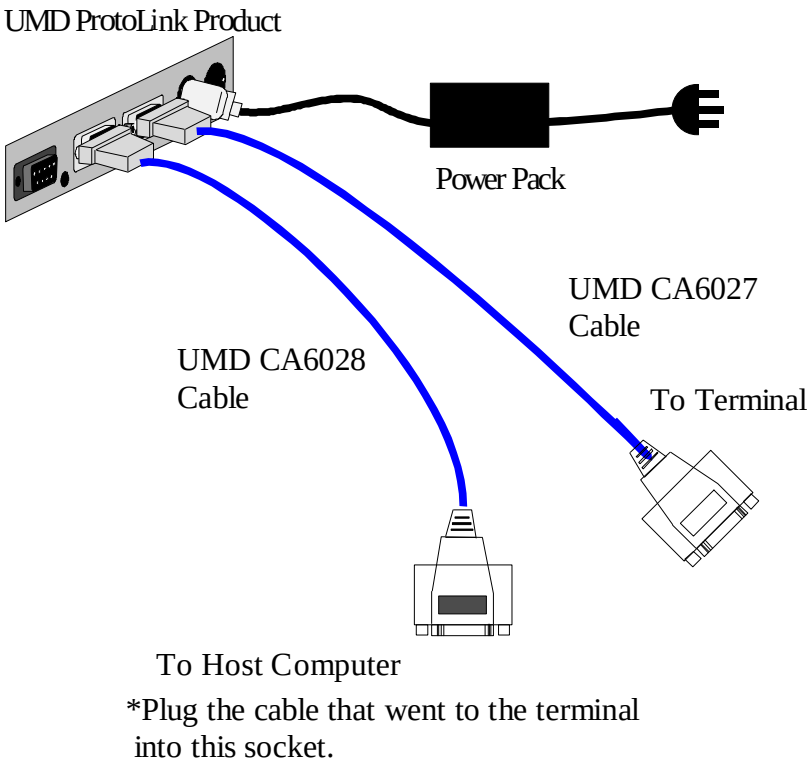


Figure 3

3 Programming and Communicating with the ProtoLink device.

The examples and explanations described in this chapter are executed with the ProtoLink Device connected in serial mode (using terminal emulation software, S143 or your software application). They can also be used in keyboard mode using UMD S143 ProtoLink Programming Utility, or using software written to send commands via the keyboard port.

The Unique Micro Design S143 Programming Utility provides a complete environment for programming and testing ProtoLink Architecture products. S143 has an easy to use menu system and can be used from the command line. The S143 User Manual provides an explanation of this software package.

The ProtoLink Device can be programmed through the BCR port using a Bar Code Wand, this method uses a series of bar codes with embedded ProtoLink commands. Programming with bar codes is discussed in chapter nine.

3.1 ProtoLink Command Syntax and Editing keys

Once in the command state, commands, data and comments are sent as lines of text which are terminated by control codes, either :

a carriage return (<CR>, $0D) ie ↵,

a line feed (<LF>, $0A),

or

a carriage return, line feed sequence (<CR><LF>).

Lines must not exceed 80 characters in length.

The command is executed upon receipt of the terminating control code.

Two simple editing keys are provided for command entry, which are usually used when typing commands from a terminal or computer serial port.

To “cancel” a line before the terminating control code,
the escape character (<ESC>, $1B) is used.

To “delete” the last character entered,
the back space character (<BS>, $08) is used.

3.2 Command Format

ProtoLink Commands, listed in Chapter 5, are ASCII characters followed by optional data values. The individual elements are delimited by a space, <TAB> or comma character.
The command can be followed by a comment, which is preceded by a semi colon (;).

ProtoLink Command [delimiter] data ; comment

Data is represented as:
ASCII decimal numbers (10),
ASCII hexadecimal numbers ($A5),
Strings of text, **enclosed in double quote marks** (“ABC”),
or
Device names (S1).

3.3 Character Strings

Strings of text which are enclosed between double quote marks can use the backslash ('\') character to introduce *escape sequences,* which are used to represent certain ASCII characters.

Sequence	Value	Character	Description
\a	$07	<BEL>	Audible bell
\b	$08	<BS>	Backspace
\f	$0C	<FF>	Form feed
\n	$0A	<LF>	Line feed
\r	$0D	<CR>	Carriage return
\t	$09	<HT>	Tab
\v	$0B	<VT>	Vertical tab
\\	$5C	\	Backslash
\"	$3F	"	Double quote

String escape sequences

For example : “\fHELLO\r\n”
represents; a form feed, the characters H E L L O, a carriage return and a line feed.

“LEFT\\TURN”
represents; L E F T \ T U R N

“COKE\r”
represents the string C O K E carriage return.
this can also be written as;
“COKE” $0D

3.4 The Start Up String and Script Files

The concept of the Start Up String is to enter into memory, a series of commands that are executed when the ProtoLink device is powered. The interpreter is told where to save the information, how many bytes are going to be in the string (the length attribute byte) and then each line is entered so that it will be a complete command.

The Start Up String is written to the configuration memory by sending commands to the controller via its input ports, example :

^B

To activate the interpreter send the Command State Prefix.

EE $0000 $01 $00

puts the value $01 into configuration memory address $0000 (msb) and $00 into configuration memory address $0001 (lsb). This is the address in memory for the start up string.

EE $0100

Start entering data at $0100

23

The length attribute byte

“LI S1 S2” $0D
“LI S2 S1” $0D
“DG 2” $0D

Linkages are set up that differ from the default, (other commands can be entered here as well) each command is terminated by a carriage return ($0D). This is the carriage return that is executed at the end of each command when the Start Up String is activated.

^C

The Command State Suffix is sent to release the command interpreter.

The length attribute byte for the start up string is calculated by counting the characters within the quotation marks plus the carriage return ie:

“LI S1 S2” $0D
is a total of 9 items for this line,

The number of items for each line are added together to calculate the length attribute byte, for example:

“LI S1 S2” $0D
“LI S2 S1” $0D
“DG 2” $0D

is a total of 23 items.

The Start Up String can be entered into the ProtoLink device, using the correct syntax, one line at a time. For example :

```
^B      ↵
EE $0000 $01 $00  ↵
EE $0100      ↵
23      ↵
“LI S1 S2” $0D        ↵
“LI S2 S1” $0D        ↵

        etc
```

But for simplicity a text file (**Script**) is written containing the commands and values to be programmed into the EEPROM. Comments explaining the configuration can also be added.

Example taken from UMD Example1.SCR

```
^B
; Start up script
; --------------
EE $0000 $01 $00          ; start of start up script is $0100
EE $0100                  ; start entering start up string from here
27                        ; string length of start up string
"LI S1 S2" $0D            ; S2 echoes S1 input
"LI S2 S1" $0D            ; S1 echoes S2 input
"LI L0 S1" $0D            ; all keypad, BCR, mag card data to S1
^C
```

Script files are sent to the ProtoLink device using S143, via the keyboard or serial ports. S143 is also used to create the information to download to the protolink device without going through the above proceedure.

3.5 Script Files and Programming the Keypad Matrix.

Script files are natively used to program the keypad matrix, even though, it is easier and quicker to program the keypad using the Programming Utility S143. The key values that are programmed in the script file are listed in the appendix.

To programming the keypad matrix a pointer is set up to tell the processor where to find the “Keypad Definition”. The keypad definition consists of a “keyboard layer header” followed by “key definition data” that is the table which contains the values for each key.

Multiple keyboard layers must be placed directly after each other with a null (0) keyboard layer header to indicate no more layers.

Setting up the pointer in a script file:

```
; Set up pointer to keypad definition
; --------------------------------
EE $0088 $02 $00        ; start of keypad definition is $0200
EE $0087 $01            ; base keyboard layout number
```

The keyboard layer header describes the keyboard layer ID (1 to 127), the number of keys (must be a multiple of eight) and an indicator LED number (0 - 16). A zero value indicates no associated LED.

```
1       ; KEYBOARD_ID:    (byte) unique keyboard identifier
128     ; NR_KEYS:        (byte) number of keys in layout
1       ; LED_NR:         (byte) LED number to light on kbd selection
```

The key definition data has two variable length strings associated with each key. The first is the “MAKE” string which is output upon pressing the key, the second is the “BREAK” string which is output upon releasing the key. Each string is preceded by a “length byte” telling the processor how many bytes in the string, example:

```
2, 0, 122           ; <alt> 3, make value, 2 bytes
0                           ; break value, no bytes
```

The standard UMD M264 ProtoLink keyboard contains 128 keys, in 16 columns each of 8 keys. Therefore each column is described with 16 lines of data, (the make value followed by the break value). To provide readablity, a comment is written before the data of each column, indicating the column number, A - P.

On the next page is a part of a keypad matrix script.

```
; Set up pointer to keypad definition
; --------------------------------
EE $0088 $02 $00        ; start of keypad definition is $0200
EE $0087 $01            ; base keyboard layout number

EE $0200                    ; start of keypad definition,
; the information for the keyboard layers and data is entered here.

; Keyboard Layout #1

    1       ; keyboard ID
    128     ; number of keys
    1       ; LED number

; Column A
1, $1B              ; make value (ESC)
0                   ; break value

            ≈
            ≈

; Column P
2, 0, 122           ; <alt> 3
0

            ≈
            ≈

; Keyboard layout #2
; ------------------

2                   ; keyboard ID
128                         ; number of keys
2                   ; LED number

; Column A
1, $1B              ; make value (ESC)
0                   ; break value

            ≈
            ≈

; Column P
2, 0, 122           ; <alt> 3
0

            ≈
            ≈

; Keyboard Layout #0 - End of keyboard chain
; ------------------
0                   ; keyboard ID of zero indicates end of chain
```

Reset is required so that the data is written into RAM and becomes active.

The ProtoLink device can be programmed to accept any of the layers as the " Base " layer, that is, the layer that is active on reset. The "keyboard ID" number is placed in memory;

```
EE $0087 $02          ; base keyboard layout number
```

Keyboard layer 2 is now the active layer on reset.

3.6 Keypad Layers and Keypad Commands.

The ProtoLink keyboards can have up to 127 user programmable layers. Each layer can be viewed as a separate keyboard, for example in a restaurant application, layer one, might represent the lunch menu and layer two, the dinner menu.

Layer numbers 128 to 255 are allocated to defined permanent layers which are held in memory.

Documented on the next page are, layer 128, which is the factory default layout and layer 129, a selectable layer.

There are a group of ProtoLink commands associated with the keypad. These can be issued directly to the command interpreter or used in a script file to control a key function or change layers.

Keypad Layer Commands	
KD	Deselect current keyboard layout.
KH i	Hold keyboard layout **i**. Temporarily locks layout **i** until after a second key is pressed.
KL i	Lock keyboard layout **i**. Makes the specified layout the locked layer, KU unlocks.
KS i	Select keyboard layout **i** (KD deselects layer).
KU	Unlock current keyboard layout.
Other Keypad Commands	
KM i j	Key Mode. Where **i** = mode; **0** never pressed, **1** no auto repeat, **2** auto repeat rate A, **3** auto repeat rate B. **j** = key number (**0** = all keys).
KP i	Press (ie make) key number **i**.
KR i	Release (ie break) key number **i**.
KB i	Make keyboard layout **i** the Base layout temporarily (EE location $87 is base layout on power up).

ProtoLink Commands associated with the keypad

Permanently defined keyboard Layer number 128 returns two characters for each key, which represents the row/column key position

Row	Column															
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**a1**	**b1**	**c1**	**d1**	**e1**	**f1**	**g1**	**h1**	**i1**	**j1**	**k1**	**l1**	**m1**	**n1**	**o1**	**p1**
2	**a2**	**b2**	**c2**	**d2**	**e2**	**f2**	**g2**	**h2**	**i2**	**j2**	**k2**	**l2**	**m2**	**n2**	**o2**	**p2**
3	**a3**	**b3**	**c3**	**d3**	**e3**	**f3**	**g3**	**h3**	**i3**	**j3**	**k3**	**l3**	**m3**	**n3**	**o3**	**p3**
4	**a4**	**b4**	**c4**	**d4**	**e4**	**f4**	**g4**	**h4**	**i4**	**j4**	**k4**	**l4**	**m4**	**n4**	**o4**	**p4**
5	**a5**	**b5**	**c5**	**d5**	**e5**	**f5**	**g5**	**h5**	**i5**	**j5**	**k5**	**l5**	**m5**	**n5**	**o5**	**p5**
6	**a6**	**b6**	**c6**	**d6**	**e6**	**f6**	**g6**	**h6**	**i6**	**j6**	**k6**	**l6**	**m6**	**n6**	**o6**	**p6**
7	**a7**	**b7**	**c7**	**d7**	**e7**	**f7**	**g7**	**h7**	**i7**	**j7**	**k7**	**l7**	**m7**	**n7**	**o7**	**p7**
8	**a8**	**b8**	**c8**	**d8**	**e8**	**f8**	**g8**	**h8**	**i8**	**j8**	**k8**	**l8**	**m8**	**n8**	**o8**	**p8**

Permanent defined keyboard layer number 129 provides a QWERTY style of keyboard. This layer also has three keys programmed to provide access to three other layers, these keys are “L1”, “L2” and “L3”. The “sp” character is the space key, “bs” is back space and “Ent” is enter (carriage return).

Row	Column															
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1														**L1**	**L2**	**L3**
2																
3																
4		**!**	**@**	**#**	**$**	**%**	**^**	**&**	*****	**(**	**)**	**_**	**=**	**/**	**+**	**-**
5		**Q**	**W**	**E**	**R**	**T**	**Y**	**U**	**I**	**O**	**P**	**[**	**]**	**7**	**8**	**9**
6		**A**	**S**	**D**	**F**	**G**	**H**	**J**	**K**	**L**	**:**	**“**	**bs**	**4**	**5**	**6**
7		**Z**	**X**	**C**	**V**	**B**	**N**	**M**	**,**	**?**	**;**	**`**	**Ent**	**1**	**2**	**3**
8		**sp**	**sp**	**sp**	**sp**	**sp**	**sp**	**sp**	**sp**	**‘**	**<**	**>**	**Ent**	**0**	**.**	**Ent**

The base layer may be changed during operation by way of the *base-layer key (KB)*, which overrides the stored base layer number configuration parameter.

The *lock-layer key (KL)* selects a layer for use until overridden by another *lock-* or *unlock- layer key (KU)*. It operates much like a "caps lock" key on a standard keyboard.

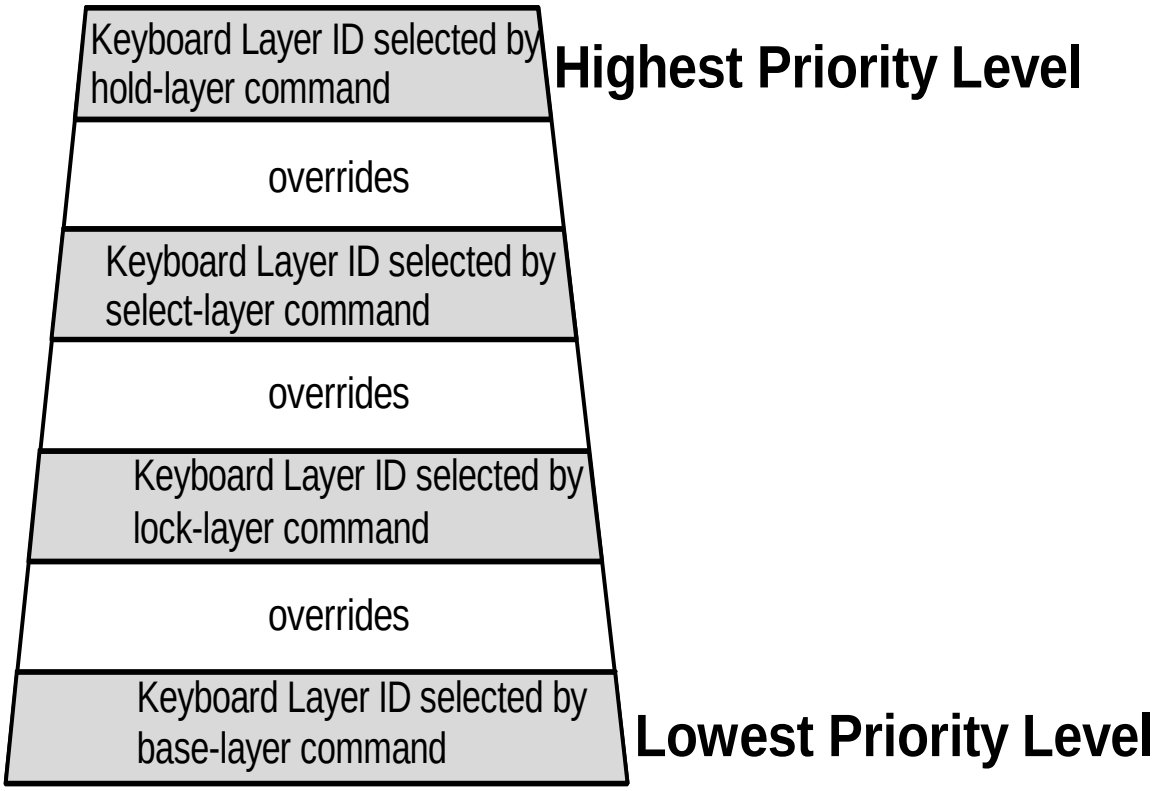
The *select-layer key (KS)* temporarily overrides any current lock or base layer and is only operationally *whilst the select-layer key is pressed.* It operates much like the shift, control or alt keys on a standard keyboard.

The *hold-layer key (KH)* temporarily locks into a layer up until a second key is pressed. Once the second key is pressed the layer that was current before the hold is reverted to. A hold overrides any selected, locked or base

3.6.1 Keypad Layer Priority

The current (active) keyboard layer is determined by a hierarchy of levels. Each level contains a value which indicates the keyboard layer that has been selected, “0” means no layer selected for this level, valid values are 1-255.

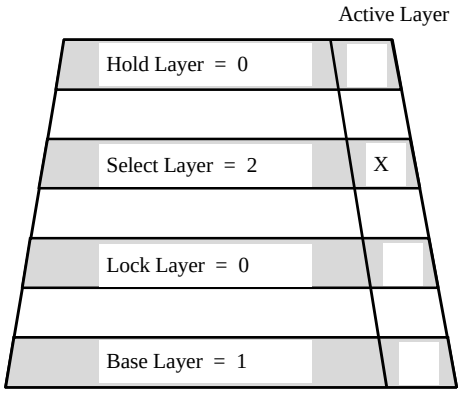
A keyboard layer that was selected by a hold-layer command overrides a layer selected by the select-layer command and so on as described in the



Starting off with the base keyboard layer number set to “1”, issuing keypad layer commands changes the active (ie the current) layer. The keypad commands can be sent to the command interpreter either from a key press or from a port. The command format is the same, example:

^B KS 2 ^C
Select keyboard layer with ID “2”

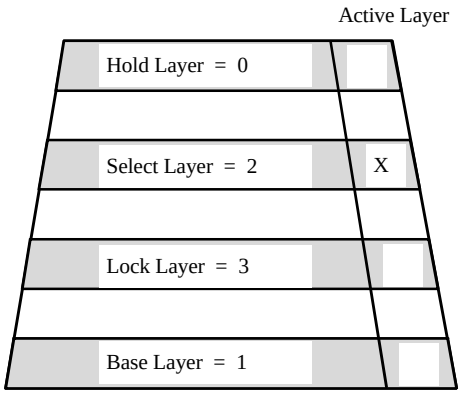
This command places “2” into the select-layer level and keyboard layer “2” becomes the active layer.



Sending the command

^B KL 3 ^C
Lock keyboard layer with ID “3”

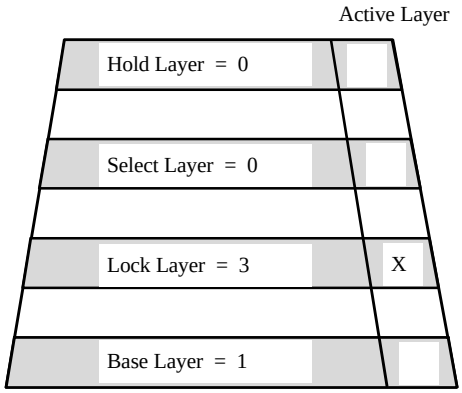
This command places “3” into the lock-layer level but dose **NOT** change the active layer, keyboard layer “2” stays the active layer because it has a higher priority level.



Sending the command

^B KD ^C
Deselect the select-layer

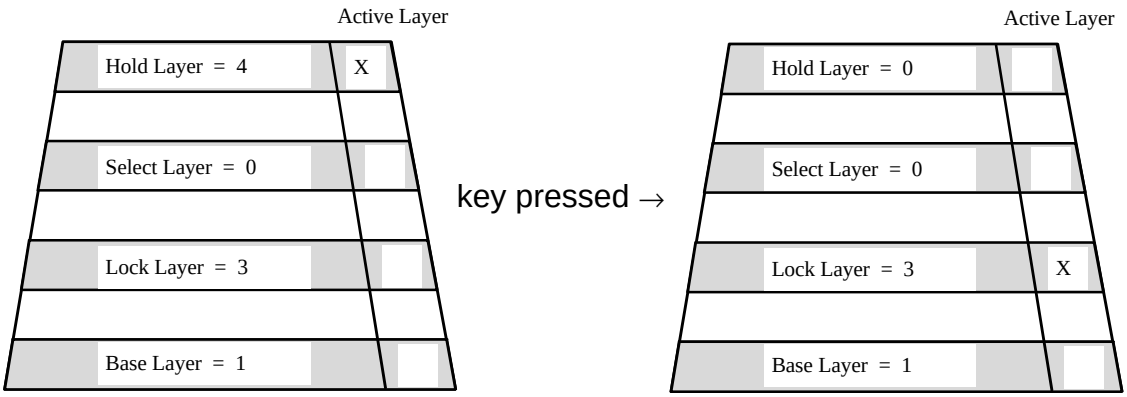
This command places “0” into the select-layer level, this was the active layer. The active layer now becomes the lock layer because it has a keyboard layer ID other than “0” and is the next priority level.



Sending the command,

^B KH 4 ^C
Hold keyboard layer with ID “4”

This command places “4” into the hold-layer level. This layer becomes the active layer .. **until** ..a key is pressed, then the next lower level with a keyboard layer ID becomes the active layer. If the key is programmed with a break value this data will be sent regardless of the layer that is now active.

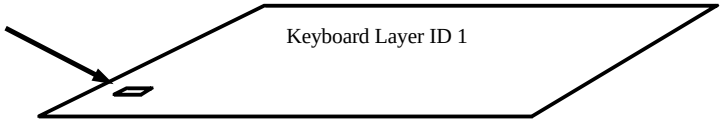


3.6.2 Keypad programming for Shift and Caps-Lock

Keys are programmable to function like the “Shift” and “Caps Lock” on a 101 keyboard. Note that this is not necessary when the ProtoLink device is connected to a computer via the H1 (host computer port) as the computer knows when the “Shift” and “Caps-Lock” keys are pressed.

One example of programming the keys for “Shift” uses the KS command

make: ^B KS2 ^C
break: ^B KD ^C

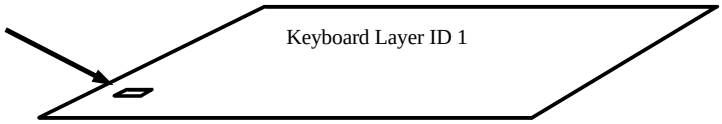


make:
break:

When the key in layer with ID 1 is pushed the command interpreter changes the active layer to layer with ID 2. The first key is still held pushed and hence the break code has not been activated. Any key in the layer ID 2 can be pushed but when the held key is released the break code is acted upon and deselects the select layer.

An example of a “Caps-Lock” key uses the KL command.

make: ^B KL2 ^C
break:



make: ^B KU ^C
break:

When the key in layer with ID 1 is pushed the command interpreter changes the active layer to layer with ID 2. The key is released and the active layer remains the layer with ID 2. Push this key again and the command interpreter makes the active layer the layer with ID 1. This is a toggle action similar to a “Caps-Lock” key.

3.6.3 Using other Keypad Commands

The custom keyboards in the ProtoLink Family have up to 128 physical keys. There are another 8 key locations that can be used for special purposes, such as a keylock or a key that is activated under program control.

The keypad commands discussed in this section are :

KP, Press (ie make) key number
KR, Release (ie break) key number

The table below indicates the location of the keys and the associated

1	9	17	25	33	41	49	57	65	73	81	89	97	105	113	121	129
2	10	18	26	34	42	50	58	66	74	82	90	98	106	114	122	130
3	11	19	27	35	43	51	59	67	75	83	91	99	107	115	123	131
4	12	20	28	36	44	52	60	68	76	84	92	100	108	116	124	132
5	13	21	29	37	45	53	61	69	77	85	93	101	109	117	125	133
6	14	22	30	38	46	54	62	70	78	86	94	102	110	118	126	134
7	15	23	31	39	47	55	63	71	79	87	95	103	111	119	127	135
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136

Sending the command,

^B KP 129 ^C

will press the key at location 129. These commands can be used to press any key under program control. It is important that the “KR” command be issued to release the pressed key.

A practical example here is the keylock switch for a supervisor layer. A keylock is wired into the hardware and represents key 129, the keyboard is programmed with 2 layers. In layer with ID 1, key 129 is programmed with:

Make: “SUP” ^B KL 2 ^C
Break: ^B KU ^C “OPT”

When the key is turned the layer with ID 2 becomes active, and when the key turned back, layer with ID 1 is the active layer (layer with ID 2 was “unlocked”). Under software control this same key can be pressed:

^B KP 129 ^C ..... ^B KR 129 ^C

3.7 Using the Link "LI" Command

Associated with each input device (eg S1) is a character buffer, when linked to an output device, the buffer is continually flushed to the linked output device. The default linkages are listed in the appendix. The link command "**LI**" sets up the connections between input and output devices, can be used to modify the default linkages in the start up string, as discussed in section 3.4 or sent to the ProtoLink device via the serial port or using S143 software:

LI id od <CR>

Logically representing the data flow,
I/P STREAM ⇒ **I/P DEVICE** (id) ⇒ IO LINKAGE ⇒ **O/P DEVICE** (od) ⇒ **O/P STREAM**

A chain of devices can be set up by way of the link command.

LI id od <CR>
LI id2 od2 <CR>

Logically representing the data flow,
I/P STREAM ⇒ **I/P DEVICE** (id) ⇒ IO LINKAGE ⇒ **O/P DEVICE** (od) ⇒ **O/P STREAM**
I/P STREAM ⇒ **I/P DEVICE** (id2) ⇒ IO LINKAGE ⇒ **O/P DEVICE** (od2) ⇒ **O/P STREAM**

To clear the data buffer of an IO linkage, the input device is linked to the Null device (N0). To have an input device active (handshake operational) but have the data go nowhere, link it to the Null device.

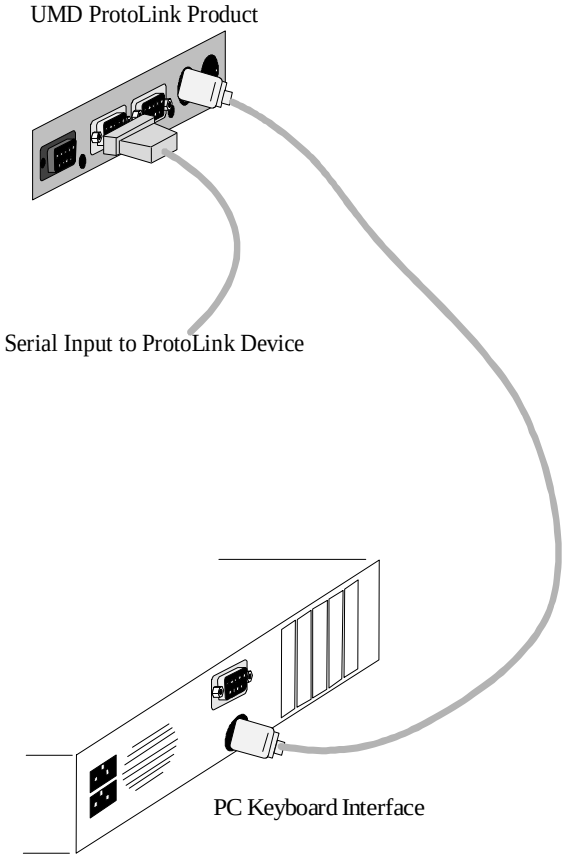
3.7.1 Link Example

For example, to have serial data into the ProtoLink device sent to a host computer (PC) as if it was typed on a keyboard.

The serial ASCII data is converted to PC/AT scan codes using the "x0" character filter device.

Data from S1 is sent to x0 and then x0 is sent to H1. The following commands set the links up chain through a number of devices:

^B LI S1 x0 ↵
LI x0 H1 ↵
^C



This can be logically represented as the following chain:

I/P STREAM (eg scanner) ⇒ S1 ⇒ x0 ⇒ H1 ⇒ O/P STREAM to PC keyboard port

3.8 The Logical Devices “L0-L3”

The logical in/out devices “**L0**” to “**L3**” have a “many in / one out” connectivity. The default linking of devices uses **L0** as the **standard input/output device**, that is; “B1”, “K1”, “M1”, “S1” and “S2” are all linked to L0 and L0 is then linked to “x0”. This has the advantage of simplifying the linking commands.

The logical devices L1 to L3 operate in the same way and are available for general use.

3.9 The “N” Devices

The logical devices “N0-N2” provide special functions to direct a data stream to, “**N0**” nowhere (**Null**), “**N1**” the **current device** (ie the device that the command came from) and “**N2**” the **hold device**.

3.9.1 Using the Hold Device

The hold device is an input only device and any device linked to it is held. The linked device is released only when it is linked to another device. For example, to have a block of data held until required, the data stream is linked to a block device and the block device to N2. When the data is required the block device is linked to another device. To reset the hold, the postamble for the blocking device has the commands for the linking.

```
LI S1 b1      ; link serial port 1 to block device b1
LI b1 N2      ; link block device b1 to hold device
```

Refer to section 3.11 for an explanation of the block device. To release the block of data send the following command;

```
LI b1 S2      ; link block device to serial port 2
              ; the postamble of b1 resets the link
              ; LI b1 N2
```

3.10 Vector Devices

The vector device “V1” is used to split up the stream into three output streams from V1, V2 and V3.

For example, to monitor the data going from S1 to H1, the vector device is linked to send the data stream to the H1 port as well as the parallel port.

The ASCII data from S1 is converted to scan codes through “x0”, the data stream is directed to the vector device and to the Host Computer port H1. V2 is directed to the hex converter, then to P1 printer port. The output here is the hexadecimal equivalent to the scan codes. V3 is directed to the Null device.

```
LI c0 P1      ; hex debug o/p to printer   <CR>
LI S1 x0      ; x0 is scan code filter     <CR>
LI x0 V1      ; V1 splits input to both    <CR>
              ; host and hex filter        <CR>
LI V1 H1      ; input into V1 links to both <CR>
LI V2 c0      ; host and hex converter     <CR>
LI V3 N0      ; but nothing else.          <CR>
```

The example can be logically represented as the following chain:

I/P STREAM ⇒ S1 ⇒ x0 ⇒ V1 ⇒ H1 ⇒ O/P STREAM to PC kbd port
V2 ⇒ c0 ⇒ P1 ⇒ O/P STREAM to printer
V3 ⇒ NULL

Remember this sequence of commands can be made part of the autoexec string for operation whenever the ProtoLink device is turned on, or entered via the command interpreter for temporary operation.

3.11 Block Devices

The “Block Devices” provide a function to place a preamble and a postamble string around a block of data. There are two block devices available, “b1” and “b2”.

Input into a block device is buffered until a terminating character is received. Once this character is received, the preamble string is output followed by the buffered characters (excluding the terminating character), followed by the postamble string.

The terminating character is specified in the configuration parameters and is “CR” by default. The buffer for “b1” is 100 bytes and for “b2” is 50 bytes. Both the preamble and postamble string can be a maximum of 255 bytes.

If the incoming data exceeds the buffer size before receiving the terminating character, the buffer is output. This output is the preamble string followed by the buffered data but not the postamble string. The buffer will fill again while waiting for the terminating character.

The example on page 7-22 uses a script file to set up the start up string, to link serial port 1 through block device 2, and then to logical output device L0.

The address of the preamble and postamble strings are entered into memory which is specified in the configuration parameters. (Refer to Chapter 6 page 6-8) In this example the address of the block device preamble string is placed in memory with the line:

```
EE $0015 $01 $20        ; BlkDev2PreAddr
```

and the address of the block device postamble string is placed in memory with the line:

```
EE $0017 $01 $30        ; BlkDev2PostAddr
```

This indicates that the preamble string is found at $0120 and the post amble string at $0130. In the example “F2” is the preamble and “F7” is the postamble. Also in the example the terminating character has been set as “Line Feed” ($0A)

```
^B                  ; Enter command interpreter
; File: C:\S143\TESTBLK.SCR
; Created by S143 vers 1.5 - UMD ProtoLink Programming Utility
; Date: 24/11/95
;

EE $0000 $01 $00       ; StartupStrAddr
EE $0100             ; StartupStr
18
"LI S1 b2" $0D
"LI b2 L0" $0D

EE $0015 $01 $20       ; BlkDev2PreAddr
EE $0120             ; BlkDev2PreStr
2
$00 $3C              ; F2

EE $0017 $01 $30       ; BlkDev2PostAddr
EE $0130             ; BlkDev2PostStr
2
$00 $41              ; F7

EE $0019 $0A          ; BlkDev2Terminator


; Now Reset the Keyboard - no comments after "RE"
RE
```

The RE command is used to reset the ProtoLink device so that the data is written into RAM and becomes active.

3.12 Delay Devices

The Intercharacter Delay Devices “**d1**”, “**d2**” and “**d3**” are software devices used to delay the dataflow from one device to another. The default setting for **d1** is **10mS**, **d2** is **20mS** and **d3** is **50mS**. (See Chapter 6, page 6-7)

For example, each character from S2 is delayed by 20mS before going to S1.

```
LI S2 d2
LI d2 S1
```

3.13 Key Scan Code Filter Devices

Keyboard scan codes are the characters that keyboards use to communicate with computers and terminals on their keyboard ports. The key scan code filter devices, "**x0 - x3**" convert ASCII characters into equivalent scan codes.

The "**x0**" filter device has its personality governed by the value set in the configuration parameters, at location $0C. This can be AT scan codes, XT scan codes or no output. (Chapter 6, page 6-7)

The "**x1**" filter device converts **ASCII to XT scan codes**.

The "**x2**" filter device converts **ASCII to AT scan codes**.

The "**x3**" filter device converts **ASCII to IBM3197 terminal keyboard scan codes**.

The key board scan code filter x0 is linked when the ProtoLink product is used to send serial data to the the host computer port H1 when it is connected to the keyboard port of a computer.

```
LI S1 x0
LI x0 H1
```

3.14 Character Filter Devices

The Character filter devices "**c0 - c2**" convert each character in the data stream. The filter device **c0** is a **binary to hexadecimal ASCII converter**, **c1** is an **upper case converter** and **c2** is an **lower case converter**.

```
LI S1 c0
LI c0 S2
```

"1Ab" into S1 is output as "314162" from S2.

```
LI S1 c1
LI c1 S2
```

"1Ab" into S1 is output as "1AB" from S2.

```
LI S1 c2
LI c2 S2
```

"1Ab" into S1 is output as "1ab" from S2.

3.15 String Conversion

The string conversion device “**c4**” provides a function to have an input string of up to 255 characters, replaced by an output string of up to 255 characters.

The following example uses a script file to set up the start up string which contains the information for the string conversion.

The address of the string conversion information is entered into configuration memory at address $38(msb)$39(lsb). The string data in this case is located at $0120.

```
EE $0120                ; c4InStr
19                      ; instring
"the quick brown fox"
12                      ; Out String
"the lazy dog"
```

The length of the input string is entered first, followed by the data. The length of the output string is next followed by its data.

```
^B              ; Enter command interpreter
; File: C:\S143\C4DEV.SCR
; Created by S143 vers 1.5 - UMD ProtoLink Programming Utility
; Date: 1/1/80
;

EE $0000 $01 $00        ; StartupStrAddr
EE $0100                ; StartupStr
18
"LI S1 c4" $0D
"LI c4 L0" $0D

EE $0038 $01 $20        ; c4InStrOutStrAddr
EE $0120                ; c4InStr
19                      ; instring
"the quick brown fox"
12                      ; Out String
"the lazy dog"

; Now Reset the Keyboard - no comments after "RE"
RE
```

When the string “the quick brown fox” is in the data stream into S1, the string output from L0 will be “the lazy dog”.

3.16 Character to string conversion

The character to string conversion device "**c5**" provides a function that allows; 1 character to be replaced by a string of up to 255 characters. The "c5" device can have a list of up to 255 characters, each with a different output string.

The address of the character conversion information is entered into configuration memory at address $3A(msb)$3B(lsb). The data is saved to memory in the form:

```
#Address of data
Input character 1
number of bytes in output string #1
output string #1 (max 255 characters)

      ≈
      ≈

Input character 255
number of bytes in output string #255
output string #255 (max 255 characters)
0
0
```

The table is always terminated with "0" in the character position and "0" in the string position.

3.17 Character Counters

The software devices "**i1**", "**i2**", and "**i3**" are used to count the number of characters in the data stream and append a string. The number of characters to count is located at $40 for "i1", $43 for "i2" and $46 for the "i3" (max count is 255 characters). The addresses of the associated output string is shown in the table in Chapter 6 page 6-18. The script file below shows the configuration required to setup the character counter.

```
^B                  ; Enter command interpreter
; File: C:\S143\ICOUNT.SCR
; Created by S143 vers 1.5 - UMD ProtoLink Programming Utility
; Date: 5/1/80
EE $0000 $01 $00        ; StartupStrAddr
EE $0100              ; StartupStr
18
"LI S3 i2" $0D
"LI i2 L0" $0D

EE $0044 $01 $20        ; DownCountStr2Addr
EE $0120              ; DownCountStr2
7                     ; number of characters in output string
"the end"             ; output string

EE $0043 $0A          ; DownCharCounter2, count 10 characters

RE
```

3.18 Control Commands

The control commands **CB**, **CL** and **CC** provide control for the internal buzzer, the LED indicators and the cash drawer output port. The following table outlines the command structure for each of the hardware devices.

Command	Description
C{a} i j	Control device type "a" given by bit mask i with action code j. eg CL $8001 2
where device type a	= "L" for LED = "B" for Buzzer = "C" for Cash drawer
where action code j	= "0" for turn OFF (default if no action code provided) = "1" for turn ON = "2" for TOGGLE = "3" for PULSE (one shot) = "4" for FLASH

3.18.1 Control Buzzer

The control buzzer "**CB**" command is used to control the internal buzzer. The buzzer can be turned "on":

```
^B ↵
CB 1 1 ↵
^C ↵
```

The buzzer can be turned "off":

```
^B ↵
CB 1 0 ↵
^C ↵
```

Other controls for the buzzer are:

```
CB 1 2      ; Toggle ( on to off, off to on )
CB 1 3      ; Pulse (one shot), turn buzzer on for the time period set
              by the flash rate controlled by the configuration parameter at
              address $0A, 0.5 second by default. If buzzer is on, it will turn
              off for this time period.
CB 1 4      ; Pulse ( on/off at flash rate )
```

3.18.2 Controlling the LED Indicators

The control led **"CL"** command is used to control the state of the LED indicators. The LED's can be turned "Off", "On", "Toggled", "Pulsed" or set "Flashing"

The 16 LED's are mapped using a sixteen bit hexadecimal number ($AAAA), the left hand LED (LED 1) is the least significant digit (LSB) and the right hand LED (LED 16) is the most significant digit (MSB). This is represented in the form.

MSB	16,15,14,13	12,11,10,9	8,7,6,5	4,3,2,1	**LSB**
Right Hand LED	0000	0000	0000	0000	Left Hand LED

To operate LED number 8, the bit map for this LED is **"0000000010000000"** , $0080 in hexadecimal.

Turning the LED "ON":

```
^B ↵
CL $0080 1 ↵
^C ↵
```

Setting the LED for "Flash" mode:

```
^B ↵
CB $0080 4 ↵
^C ↵
```

Other controls for the LED's are:

```
CB $0080 2  ; Toggle ( on to off, off to on )
CB $0080 3  ; Pulse (one shot), flash LED on (if off) flash LED off
              (if on) for the time period set by the flash rate controlled by
the configuration parameter at address $0A, 0.5 second by
default.
CB $0080 0  ; Turn off, if already off, no action occurs
```

3.18.3 Operating the Cash Drawer output

The control cash drawer "**CC**" command is used to control the cash drawer solenoid driver. There are 2 ports optionally available.
The output is activated by the command:

^B ↵
CC 1 1 ↵
^C ↵

or:

^B ↵
CC 2 1 ↵
^C ↵

Other action codes (0, 2-4) have the same effect.

3.19 Dump Global variable command

The Dump Global "**DG**" command outputs the specified global variable as an ASCII string followed by a carriage return.

Command	Description
DG i	Dump Global variable i

The following table lists the available global variables.

Number	Description
1	IO linkages
2	Firmware version
3	Firmware module revisions

3.19.1 List I/O Linkages

Dump Global variable 1 list the I/O linkages for each device which is not linked to the Null device. The format is:

input device -> output device

^B ↵
DG 1↵ (ProtoLink Device responds with)
L0->S1 S1->L0 S2->L0 M1->L0 B1->L0 H1->E1 E1->H1 K1->L0 x0->H1

^C ↵

3.19.2 Display Firmware Version

Global variable two displays the firmware revision of the EPROM on the micro controller module:

^B ↵

DG 2↵ (ProtoLink Device responds with)
M301-F00 Firmware P/N 7-0002-109-5
13/02/95
(C) Copyright Unique Micro Design P/L
Melb, Australia

^C ↵

3.19.3 Display Firmware Module revisions

Global variable three displays the firmware revisions of the various software modules in the EPROM on the micro controller module:

^B ↵

DG 3↵ (ProtoLink Device responds with)
118VECS 21A/07/92 118DEF 14A/04/92 START520 19A/09/91
H8BUF 18A/07/91 H8SER 27A/09/93 H8UTIL 21A/07/92
H8STR 12A/01/94 H8DEBUG 27A/09/93 H8MTESR 17A/07/91
H8MT18A/08/93 GENIO 15A/12/92 H8DEVICE 29A/08/94
118SCODE 06A/04/93 101KBD 16A/01/95 118IO 13A/01/94
118LCD 22A/03/93 118KEY 22A/07/93 118MCR 27A/04/92
118BCR 13/02/95 118TM 23A/05/94 118GILB 31A/03/93
118CLK 20A/01/94 118KBD 19A/02/93 118EEPRM 14A/05/93
118CI 19A/01/94 ASC2COMP 21A/02/94 118TASKS 17/01/95
118EXP 08A/01/93 118PROJ 10A/05/93 118CONFG 17/01/95
M301-F00 Firmware P/N 7-0002-109-5
13/02/95
(C) Copyright Unique Micro Design P/L
Melb, Australia
^C ↵

3.20 Reset EEPROM to Default Values

The reset "**RD**" command is used to reset the EEPROM to the default values from the EPROM. This is used in combination with the "**RE**" command which resets the ProtoLink device.

^B ↵
RD↵
RE↵
^C ↵

3.21 Delay command

The delay "**DL**" command is used to create a delay in the current operation. Commonly used as part of a keypad key definition.

The following example is an extract from a script file showing the DL command delaying each letter in the word "help" (by 1 second) when the key is pressed:

```
; Column A
35, "H" $02 "DL100" $0D $03 "E" $02 "DL100" $0D $03
            "L" $02 "DL100" $0D $03 "P" $0D          ; make value
0                   ; break value

            ≈≈
            ≈≈

; Column P

            ≈≈
            ≈≈

; Keyboard Layout #0 - End of keyboard chain
; -----------------
0                   ; keyboard ID of zero indicates end of chain
```

3.22 Put Device command

The put device "**PD**" command is used to send bytes of data to a specified device. The following example sends seven bytes of data (a form feed control code, "HELLO" and a line feed control code to the ProtoLink display (device D1):

```
^B ↵
PD D1 $0C "HELLO" $0A ↵
^C ↵
```

3.23 Memory Commands

Memory commands are a group of commands that are used to view and edit the memory areas in the ProtoLink Architecture. These areas are EEPROM (non-volatile memory) "**E**", RAM "**R**" and IOBUS (Peripheral Interface Bus) "**I**".

These commands are used for diagnostic purposes and to manipulate memory and the modules connected to the Peripheral Interface Bus.

Command	Description
E{a} i j...	Enter bytes j... to memory type "a" starting from address i
B{a} i j k	Set bits indicated by bit mask j and clear bits indicated by bit mask k for memory type "a" at address i.
D{a} i [j]	Dump j bytes from memory type "a" starting from address i.
O{a} i [j]	Output bytes from memory type "a" starting from address i. If j = 0 or not provided then the bytes are output until a zero (null) character is found. If parameter j is entered then j bytes are output.
where memory type a	= "R" for RAM = "E" for EEPROM = "I" for IOBUS

Format of the Memory Commands

3.23.1 Enter Bytes Command

The Enter Bytes commands "**EE**", "**EI**" and "**ER**" can accept a string of data with the first character indicating the number of bytes and terminated with a <CR>.

Many examples of this type of command are shown in sections 3.4 - 3.6 of this chapter.

3.23.2 Bit Mask Command

The Bit Mask commands "**BE**", "**BI**" and "**BR**" are used to set specific bits (make them "1") and clear specific bits (make them "0") without effecting the remaining bits at that address.

For example, changing the value in the configuration memory (EEPROM) for the decoding of bar code symbologies. Only two symbologies need to be changed (without effecting the other settings), APN/UPC to be enabled and Code 39 disabled. (See Chapter 6 section 3.7.2).

Command	Address	Set these bits to "1"	Clear these bits to "0"
BE	$0023	(00010000) $10	(00000010) $02

The above table shows the bit mask to be set and cleared, note that a "1" clears the bit, to a value of "0".

^B ↵
BE $0023 $10 $02 ↵
^C ↵

3.23.3 Output Bytes command

The Output bytes commands "**OE**", "**OI**" and "**OR**" display the contents of the memory, the output is unformatted hexadecimal.

^B ↵
OE $0100 10 ↵ Output 10 bytes starting from address $0100
09 4C 49 20 4C 30 20 53 31 0D
response from ProtoLink Device
^C ↵

3.23.4 Dump Bytes command

The Dump bytes commands "**DE**", "**DI**" and "**DR**" display the contents of the memory, the output is formatted displaying the address in hexadecimal, the data in hexadecimal and the ASCII equivalent data.

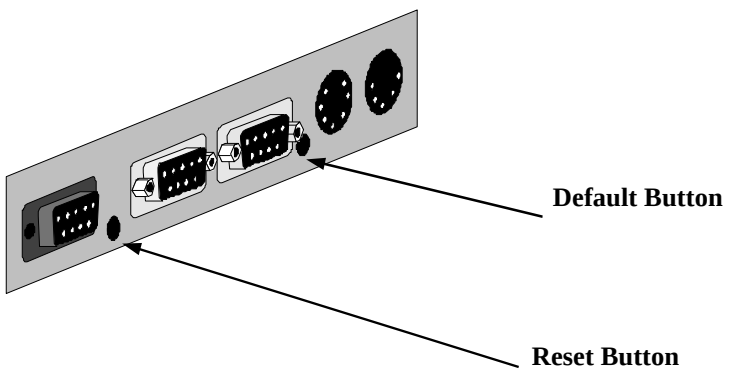
^B ↵
DE $0100 16 ↵ Output 16 bytes starting from address $0100
0100 09 4C 49 20 4C 30 20 53 31 0D 4C 49 20 62 32 20 .LI L0 S1.LI B2
response from ProtoLink Device
^C ↵

3.24 Reset and Default Buttons

The reset button physically resets the microcontroller, forcing it to perform a power up sequence.

The default button is used to return the non volatile configuration memory back to the default settings. To use this facility, the default button is held whilst pressing the reset button. Upon reset the controller module will issue three rapid beeps to indicate that the defaults have been set.

NOTE:
If any changes have been made to the programming, these will be lost when the reset to defaults sequence above is used.



Location of the Default and Reset Buttons

Unique Micro Design
ProtoLink Architecture Family of Products
30/01/96

Appendix
Application Notes, Technical notes and Technical Standards

A

The Appendix contains Application Notes, Technical Notes and Technical Standards which apply to the ProtoLink Architecture

Part Number	Title	Description	Date
Application Notes			
7-5010-128-6	AN128	Default Device Linkages	10/11/95
7-5010-125-1	AN125	Summary of the Standards required for the Physical requirements of Magnetic Cards	10/3/95
7-5010-124-8	AN124	Documentation list for the UMD ProtoLink Architecture	07/03/96
Technical Notes			
7-5020-102-3	TN102	ASCII, Extended ASCII and keyboard Scan Codes for the ProtoLink Keypad key programming	11/11/95
Technical Standards			
7-6010-119-1	TS119	S143 Initialisation File Parameters and associated Configuration Memory Addresses	17/10/95
7-6010-101-6	TS101	UMD Standard DB9-P Serial Port	10/04/95
7-6010-103-0	TS103	UMD Standard DB25 Parallel I/O Port	06/10/95
7-6010-107-8	TS107	UMD Standard 7 Pin DIN Computer Keyboard and Power Interface	10/10/95
7-6010-109-2	TS109	UMD Standard AMP DB9 BCR Wand Interface	10/10/95
7-6010-113-9	TS113	UMD Standard 5 Pin DIN Keyboard Interface	11/10/95
7-6010-118-4	TS118	UMD Standard 5 Pin DIN Cash Drawer Interface	11/10/95
7-6010-120-7	TS120	Standard DB15HD VGA Monitor Interface	20/10/95
7-6010-121-4	TS121	UMD ProtoLink Architecture Peripheral Interface	06/03/96

Application Note 128

Introduction

This application note documents the default device linkages, integrated into the ProtoLink Architecture range of firmware options. This document **does not** address any other effected attributes.

Firmware	Devices												
Number	B1	b2	d2	E1	H1	K1	L0	M1	p1	p2	S1	S2	x0
F00	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0			S1➡L0	S2➡L0	x0➡H1
F02	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡S1	M1➡L0			S1➡S2	S2➡S1	
F03	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡S1	M1➡L0			S1➡L0	S2➡L0	x0➡H1
F04	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0			S1➡D1	S2➡L0	x0➡H1
F05	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0			S1➡L0	S2➡L0	x0➡H1
F06	B1➡L0		d2➡S1	E1➡H1	H1➡E1	K1➡L0	L0➡d2	M1➡L0			S1➡S1	S2➡L0	x0➡H1
F07	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0	p1➡L0	p2➡S2	S1➡L0	S2➡p1	x0➡H1
F08	B1➡d2		d2➡L0	E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0			S1➡L0	S2➡L0	x0➡H1
F09	B1➡L0	b2➡L0		E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0			S1➡L0	S2➡b2	x0➡H1
F14	B1➡L0			E1➡H1	H1➡E1	K1➡L0	L0➡x0	M1➡L0			S1➡L0	S2➡L0	x0➡H1

List of the physical and special devices and the links that are set up as default.

Device	I/O	Description
B1	I	Bar Code Reader.
*bn	I/O	Block Device n (n=”1”..”2”)
*dn	I/O	Delay Device n (n=”1”..”3”)
E1	I	External keyboard
H1	O	Host Computer (via keyboard i/f)
K1	I	Internal keyboard
*L0	I/O	Standard I/O
M1	I	Magnetic Card Reader
pn	I/O	Gilbarco comms 2
Pn	I/O	Parallel i/o port n (n=”1”..”3”)
Sn	I/O	Serial i/o port n (n=”1”..”6”)
*xn	I/O	Key scan filter n (n=”0”..)

Description of the physical and special devices used in the previous table

Application Note 125

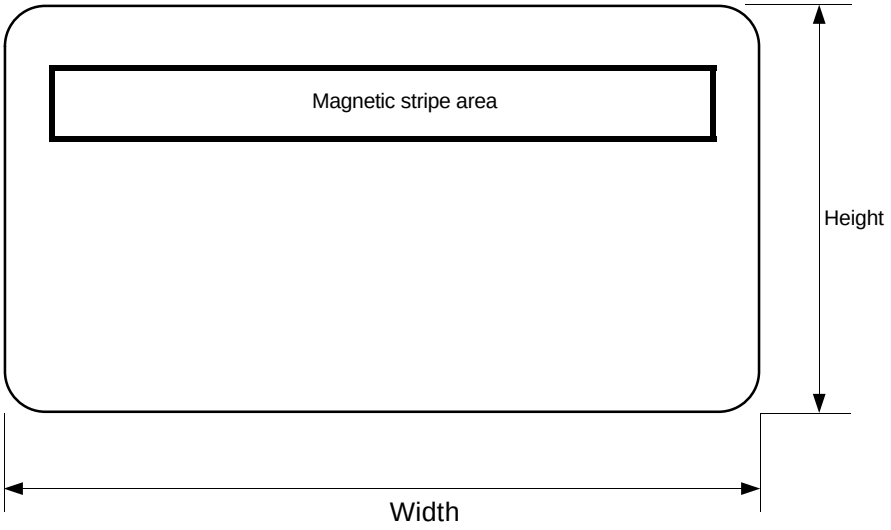
Introduction

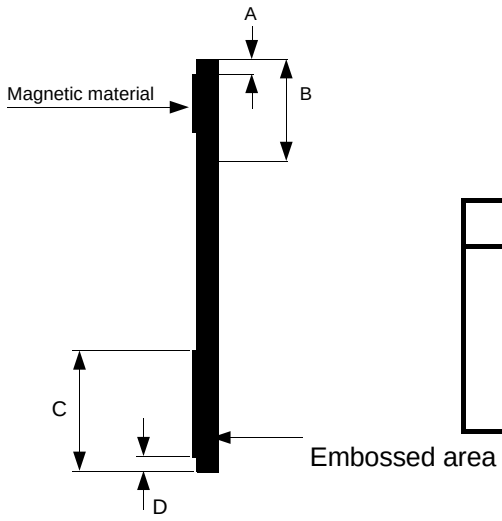
This application note is a brief summary of the standards required for the physical requirements of Magnetic Cards.

Physical requirements:

The physical requirements should conform to those in Australian Standards AS 3521 - 1988 (or ISO 7810). The nominal dimensions of the three sizes of card are shown in the table below.

Card Type	Width		Height		Thickness	
	mm	in	mm	in	mm	in
ID-1	85.60	3.370	53.98	2.125	0.76	0.030
ID-2	105	4.134	74	2.913	0.76	0.030
ID-3	125	4.921	88	3.465	0.76	0.030





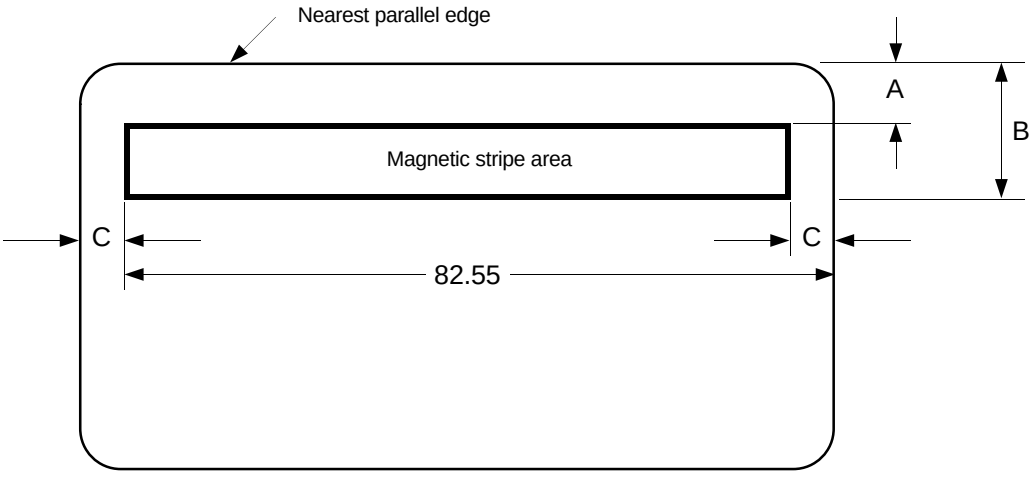
Dimension	mm	Description
A *max*	2.54	Top edge to magnetic track
B *min*	19.05	B - A distortion free area
C *max*	24.03	C - D Embossed area
D *min*	2.54	Lower edge to embossing

Track Structure

Recording techniques and structure should conform to Australian Standards AS 3524-1988 (ISO 7811).There are two specifications for the magnetic track size, as described below. The magnetic medium can contain up to 3 tracks of data. Tracks one and two are read only while track three is read/write. Track 1 is the only alpha-numeric track, tracks two and three must be numeric data only.

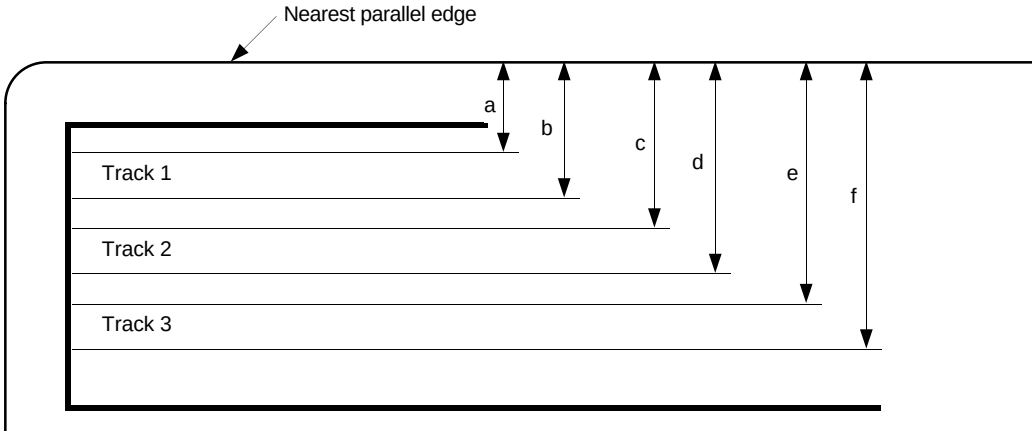
Location of tracks

The location of the tracks are all with respect to the nearest parallel edge to the magnetic strip.



Dimensions mm	Single or dual track	Three tracks
A	5.54 max	5.54 max
B	11.89 max	15.82 max
C	2.92 max	2.92 max

Location of Magnetic Tracks



	a	b	c	d	e	f
mm	5.66	8.46 - 8.97	8.46 - 8.97	11.76	12.01 - 12.52	15.32 - 15.82

Magnetic track format of common cards

Track 1 data

SS	FC	PAN	FS	NAME	FS	ADD INFO	ES
%	FORMAT CODE	19 DIGITS MAX	^	25 ALPHANUMERIC	^	VARIOUS S	?

Track 2 data

SS	PAN	FS	ADD INFO	ES	
;	19 DIGITS MAX	=	VARIOUS	?	CR

Notes:
SS - Starting String
FS - File separator
ES - End string

Unique Micro Design Pty Ltd ***(ACN 007 419 490)***
16 Nyadale Drive, Scoresby, Victoria 3179, Australia
Tel: +61-3-764-8166
Fax: +61-3-764-8177

ASCII, Extended ASCII and Keyboard Scan Codes for the ProtoLink Architecture Keypad key programming

The key definitions are in ASCII, eg “A” or single byte scan code, eg 01 or a two byte extended code, eg 0,30

Key	Shift	Ctrl	Alt
1	!		0,120
2	@		0,121
3	#		0,122
4	$		0,123
5	%		0,124
6	^		0,125
7	&		0,126
8	*		0,127
9	(		0,128
0	)		0,129
-	_		0,130
=	+		0,131
[	{		0,26
]	}		0,27
\	|		0,43
;	:		0,39
'	"		0,40
,	<		0,51
.	>		0,52
/	?		0,53

Unique Micro Design Pty Ltd (ACN 007 419 490)
16 Nyadale Drive, Scoresby, Victoria 3179, Australia
Tel: +61-3-9764-8166
Fax: +61-3-9764-8177

Key	Shift	Ctrl	Alt
a	A	01	0,30
b	B	02,02 [1]	0,48
c	C	03	0,46
d	D	04	0,32
e	E	05	0,18
f	F	06	0,33
g	G	07	0,34
h	H	08	0,35
i	I	09	0,23
j	J	10	0,36
k	K	11	0,37
l	L	12	0,38
m	M	13	0,50
n	N	14	0,49
o	O	15	0,24
p	P	16	0,25
q	Q	17	0,16
r	R	18	0,19
s	S	19	0,31
t	T	20	0,20
u	U	21	0,22
v	V	22	0,47
w	W	23	0,17
x	X	24	0,45
y	Y	25	0,21
z	Z	26	0,44

[1] Control B opens the command interpreter in the ProtoLink Device, the second control B exits the interpreter and sends the control B. This requires the code to be entered twice (02,02).

Key	Programmed code	Shift	Ctrl	Alt
Tab	09	0,15	0,148	0,165
Backspace	08		127	0,14
Enter	13		10	0,28
Esc	27			0,01
Space	32			

The following group is commonly referred to as the **grey** keys.
That is the cursor and page keys

Key	Programmed code	Shift	Ctrl	Alt
Home	0,71		0,119	0,151
Up	0,72		0,141	0,152
PgUp	0,73		0,132	0,153
Left	0,75		0,115	0,155
Right	0,77		0,116	0,157
End	0,79		0,117	0,159
Down	0,80		0,145	0,160
PgDn	0,81		0,118	0,161
Ins	0,82		0,146	0,162
Del	0,83		0,147	0,163

Function Keys.

Key	Programmed code	Shift	Ctrl	Alt
F1	0,59	0,84	0,94	0,104
F2	0,60	0,85	0,95	0,105
F3	0,61	0,86	0,96	0,106
F4	0,62	0,87	0,97	0,107
F5	0,63	0,88	0,98	0,108
F6	0,64	0,89	0,99	0,109
F7	0,65	0,90	0,100	0,110
F8	0,66	0,91	0,101	0,111
F9	0,67	0,92	0,102	0,112
F10	0,68	0,93	0,103	0,113
F11	0,133	0,135	0,137	0,139
F12	0,134	0,136	0,138	0,140

System Control Keys.

Key	Programmed code	Shift	Ctrl	Alt
Break	0,02			
Null	0,03			
Pause	0,04			
Prt Screen	0,05		0,114	
NUM Lock	0,06			
CAPS Lock	0,07			
SCROLL Lock	0,08			

Numeric Keypad Group.

Key	Programmed code	Shift	Ctrl	Alt
Keypad Enter	0,182		10	0,166
Keypad -	0,183		0,142	0,74
Keypad *	0,184		0,150	0,55
Keypad /	0,185		0,149	0,164
Keypad +	0,186		0,144	0,78
Keypad 0	0,190		0,146	
Keypad 1	0,191		0,117	
Keypad 2	0,192		0,145	
Keypad 3	0,193		0,118	
Keypad 4	0,194		0,115	
Keypad 5	0,195		0,143	
Keypad 6	0,196		0,116	
Keypad 7	0,197		0,119	
Keypad 8	0,198		0,141	
Keypad 9	0,199		0,132	
Keypad .	127		0,147	

Control/Alt/Shift Group.
These keys generate code when pushed (make) and when released (break).

Key	Programmed code
Left Ctrl make	0,173
break	0,174
Left Alt make	0,169
break	0,170
Right Ctrl make	0,175
break	0,176
Right Alt make	0,171
break	0,172
Left Shift make	0,177
break	0,178
Right Shift make	0,179
break	0,180

The following tables list the codes in Scan-Code order.
Firstly, single byte scan code

Value	Description
01	Ctrl-A
02,02	Ctrl-B
03 - 07	Ctrl-C ..... Ctrl-G
08	Ctrl-H (Backspace)
09	Ctrl-I (Tab)
10	Ctrl-J (Ctrl-Enter) and (Ctrl-Keypad Ent)
11 - 12	Ctrl-K, Ctrl-L
13	Ctrl-M (Enter)
14 - 26	Ctrl-N .... Ctrl-Z
27	Esc
32	Space
127	Ctrl-Backspace (Keypad .)

Secondly the two byte extended code, the first byte is the NULL character (0) and the second byte is the listed extended ASCII code.

Value	Description
01	Alt-Esc
02	Break
03	Null
04	Pause
05	Prt Screen
06	NUM Lock
07	CAPS Lock
08	SCROLL Lock
14	Alt-Backspace
15	Shift-Tab
16 - 25	Alt-Q,W,E,R,T,Y,U,I,O,P
26 - 28	Alt-[,], Enter

Value	Description
30 - 38	Alt-A,S,D,F,G,H,J,K,L
39 - 40	Alt-; ‘
43	Alt-\
44 - 50	Alt-Z,X,C,V,B,N,M
51 - 53	Alt-, (comma) . (full stop) /
55	Alt-keypad *
59 - 68	Function Keys F1 - F10
71	Home
72	Cursor Up
73	PgUp
74	Keypad -
75	Cursor Left
77	Cursor Right
78	Keypad +
79	End
80	Cursor Down
81	PgDn
82	Ins
83	Del
84 - 93	Function keys Shift F1 - F10 (F11 - F20)
94 - 103	Function keys Ctrl F1 - F10 (F21 - F30)
104 - 113	Function keys Alt F1 - F10 (F31 - F40)
114	Ctrl-Prt Screen
115	Ctrl-Cursor Left (Ctrl-Keypad 4)
116	Ctrl-Cursor Right (Ctrl-Keypad 6)
117	Ctrl-Cursor End (Ctrl-Keypad 1)
118	Ctrl-PgDn (Ctrl-Keypad 3)
119	Ctrl-Home (Ctrl-Keypad 7)
120 - 131	Alt-1,2,3,4,5,6,7,8,9,0,-,=
132	Ctrl-PgUp (Ctrl-Keypad 9)

Value	Description
133 - 134	F11 - F12
135 - 136	Shift-F11 - F12
137 - 138	Ctrl-F11 - F12
139 - 140	Alt-F11 - F12
141	Ctrl-Keypad 8 (Ctrl-Cursor Up)
142	Ctrl-Keypad -
143	Ctrl-Keypad 5
144	Ctrl-Keypad +
145	Ctrl-Keypad 2 (Ctrl-Cursor Down)
146	Ctrl-Keypad 0 (Ctrl-Ins)
147	Ctrl-Keypad . (Ctrl-Del)
148	Ctrl-Tab
149	Ctrl-Keypad /
150	Ctrl-Keypad *
151	Alt-Home
152	Alt-Cursor Up
153	Alt-Pg Up
155	Alt-Cursor Left
157	Alt-Cursor Right
159	Alt-End
160	Alt-Cursor Down
161	Alt-Pg Dn
162	Alt-Ins
163	Alt-Del
164	Alt-Keypad /
165	Alt-Tab
166	Alt-Keypad Enter

Value	Description
169	Left Alt (make)
170	Left Alt (break)
171	Right Alt (make)
172	Right Alt (break)
173	Left Ctrl (make)
174	Left Ctrl (break)
175	Right Ctrl (make)
176	Right Ctrl (break)
177	Left Shift (make)
178	Left Shift (break)
179	Right Shift (make)
180	Right Shift (break)
182	Keypad enter
183	Keypad -
184	Keypad *
185	Keypad /
186	Keypad +
190 - 199	Keypad 0 - 9

UMD S143 Initialisation File

This document specifies the UMD S143 ProtoLink Programming Utility initialisation (".INI") file format. Key names are described along with the associated ProtoLink configuration memory address that the key name effects.

1. Introduction

The initialisation file is a text file which holds configuration settings laid out in an easily read format.

One normally edits an initialisation file by way of the UMD S143 software. This document provides information for more experience users who may wish to edit these files directly using a text editor.

2. Format of the INI Files

```
[sectionname]
keyname=value      ; comment
```

eg

```
[Startup]
StartupStr="LI B1 S2\r"    ; Want bar code reader to output to serial port S2
```

where:

[*sectionname*]	*Sectionname* names a *section*. The enclosing brackets ([]) are required, and the left bracket must be in the leftmost column on the screen.
keyname	*Keyname* is the name of the configuration parameter to alter. It can consist of any combination of letters and digits. *Keyname* must be followed immediately by an equals sign.
value	The *value* of a configuration parameter can be an integer in decimal or hexadecimal (ie number ends with a 'h') or a quoted string, depending on the type of entry. Quoted strings may contain escape sequences, eg "\r\n" is a two character carriage return, line feed sequence. eg BlkDev2PostStr="\r\n" , a1m=<CTL-B>"Hello"
; comment	All characters after a semicolon (;) are treated as *comments* until the end of the current line.

Unique Micro Design Pty Ltd (ACN 007 419 490)
16 Nyadale Drive, Scoresby, Victoria 3179, Australia
Tel: +61-3-9764-8166
Fax: +61-3-9764-8177
Email: umd@ozemail.com.au

Section, Key Name & Default		Config Mem Address	Bit flags (0 - 7)
[Startup]			
StartupStrAddr=0h		0000 - 1h	
StartupStr=""		String located at StartupStrAddr	
[OperatingModes]			
BuzzerDuration=02h		0002h	0 - 1
CmdPorts=05h		0002h	2 - 6
LEDScan=00h		0002h	7
CmdStatePrefix1=02h		0003h	
CmdStatePrefix2=00h		0004h	
NumLockLEDNo=0Ch		0005h	
CapsLockLEDNo=0Dh		0006h	
ScrollLockLEDNo=0Eh		0007h	
GoodReadLEDNo=0Fh		0008h	
PowerLEDNo=10h		0009h	
LEDFlashRate=19h		000Ah	
[ExternalKbdInterface]			
ExternalKbdType=02h		000Bh	
[ComputerInterface]			
ComputerInterface=02h		000Ch	
[DelayDevices]			
DelayDev1=01h		000Dh	
DelayDev2=02h		000Eh	
DelayDev3=05h		000Fh	

Section, Key Name & Default		Config Mem Address	Bit flags (0 - 7)
[BlockDevices]			
BlkDev1PreAddr=0h		0010 - 11h	
BlkDev1PostAddr=32h		0012 - 13h	
BlkDev2PreAddr=0h		0015 - 16h	
BlkDev2PostAddr=34h		0017 - 18h	
BlkDev1PreStr=""		String located at BlkDev1PreAddr	
BlkDev1PostStr="\r"		String located at BlkDev1PostAddr	
BlkDev2PreStr=""		String located at BlkDev2PreAddr	
BlkDev2PostStr="\r\n"		String located at BlkDev2PostAddr	
BlkDev1Terminator=0Dh		0014h	
BlkDev2Terminator=0Dh		0019h	
[MagneticCardReader]			
MCRPreAddr=0h		001B - 1Ch	
MCRPostAddr=32h		001D - 1Eh	
MCRPreStr=""		String located at MCRPreAddr	
MCRPostStr="\r"		String located at MCRPostAddr	
SendMCRTrack=07h		001Ah	0 - 2
MCROrder123=00h		001Ah	3
MCRConcat=00h		001Ah	4
SendMCRId=00h		001Ah	6
MultiTrackOper=00h		001Ah	7

Unique Micro Design Pty Ltd ***(ACN 007 419 490)***
16 Nyadale Drive, Scoresby, Victoria 3179, Australia
Tel: +61-3-9764-8166
Fax: +61-3-9764-8177
Email: umd@ozemail.com.au

[BarCodeReader]		
BCRPreAddr=0h	001F - 20h	
BCRPostAddr=32h	0021 - 22h	
BCRPreStr=""	String located at BCRPreAddr	
BCRPostStr="\r"	String located at BCRPostAddr	
DecodeBarCode=1Fh	0023h	0 - 4
UPCOptions1=CBh	0024h	
UPCOptions2=03h	0025h	0 - 2
CodabarOptions=04h	0026h	0 - 3
TransSymId=00h	0027h	0 - 1
TransNumDig=00h	0027h	2
ITFLength1=00h	0028h	
ITFLength2=00h	0029h	
[SerialComms]		
NetworkAddr=00h	002Bh	
SerPort1Format=42h	002Ch	
SerPort1Handshk=0Ch	002Dh	
SerPort2Format=42h	002Fh	
SerPort2Handshk=0Ch	0030h	
SerPort3Format=42h	008Eh	
SerPort3Handshk=0Ch	008Fh	
SerPort4Format=42h	0091h	
SerPort4Handshk=0Ch	0092h	
SerPort5Format=42h	0094h	
SerPort5Handshk=0Ch	0095h	
SerPort6Format=42h	0097h	
SerPort6Handshk=0Ch	0098h	

[CharFilters]		
c4InStrOutStrAddr=0h	0038 - 39h	
c4InStr=""	Data located at c4InStrOutStrAddr	
c4OutStr=""	Data located at c4InStrOutStrAddr	
c5InCharOutStrAddr=0h	003A - 3Bh	
	c5 table generated by S143 software, when c5 is used.	
[TouchMemory]		
TM1PreAddr=0h	003C - 3Dh	
TM1PostAddr=32h	003E - 3Fh	
TM1PreStr=""	String located at TM1PreAddr	
TM1PostStr="\r"	String located at TM1PostAddr	
[CharCounters]		
DownCountStr1Addr=0h	0041 - 42h	
DownCountStr2Addr=0h	0044 - 45h	
DownCountStr3Addr=0h	0047 - 48h	
DownCountStr1=""	String located at DownCountStr1Addr	
DownCountStr2=""	String located at DownCountStr2Addr	
DownCountStr3=""	String located at DownCountStr3Addr	
DownCharCounter1=01h	0040h	
DownCharCounter2=01h	0043h	
DownCharCounter3=01h	0046h	
[Monitor Display]		
MonitorFlags=00h	0049h	
MonitorEmulation=00h	004Ah	

Unique Micro Design Pty Ltd ***(ACN 007 419 490)***
16 Nyadale Drive, Scoresby, Victoria 3179, Australia
Tel: +61-3-9764-8166
Fax: +61-3-9764-8177
Email: umd@ozemail.com.au

[Custom Tasks]		
CustomTaskPtr1=0h	0050 - 51h	
CustomTaskPtr2=0h	0052 - 53h	
CustomTaskPtr3=0h	0054 - 55h	
CustomTaskPtr4=0h	0056 - 57h	
CustomTaskPtr5=0h	0058 - 59h	
CustomTaskPtr6=0h	005A - 5Bh	
CustomTaskPtr7=0h	005C - 5Dh	
CustomTaskPtr8=0h	005E - 5Fh	
[KbdModes]		
KbdLayerAddr=0h	0088 - 89h	
KeyClick=01h	0080h	0
KeyAutoRepRate=01h	0080h	6 - 7
RateADelay=25h	0081h	
RateARepRate=19h	0082h	
RateBDelay=19h	0083h	
RateBRepRate=05h	0084h	
BaseKbdLayerNo=80h	0087h	
[ConversionDevices]		
x4TablePtr=0h	008Ah	not implemented
[ParallelPorts]		
ParPort1Dir=01h	008Ch	0
ParPort2Dir=01h	008Ch	1
ParPort3Dir=01h	008Ch	2
[Defaults]	**These should not be altered**	
SizeCarrRetStr=01h	0032h	
CarrRetChar1=0Dh	0033h	
SizeCRLFStr=02h	0034h	
CarrRetChar2=0Dh	0035h	
LineFeedChar=0Ah	0036h	

[KbdLayer1]		
ID=	Data generated by the S143 software when keyboard layers are programmed.	
Nokeys=		
LEDNo=		
a1m=""		
a1b=""		
....		
p8m=""		
p8b=""		
[KbdLayer2]		
ID=		
NoKeys=		
LEDNo=		

16 Nyadale Drive, Scoresby, Victoria 3179, Australia
Tel: +61-3-9764-8166
Fax: +61-3-9764-8177
Email: umd@ozemail.com.au

UMD Standard DB9-P Serial Port

This document defines the UMD Standard DB9-P Serial Port, which provides five volts for powering peripheral devices such as bar code scanners..

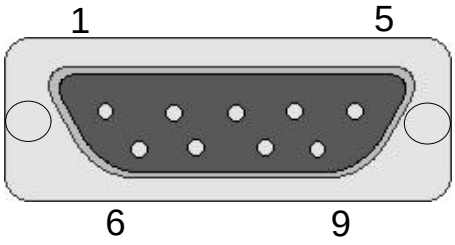
1.1 Pin Assignment

Pin assignments are a subset of the DB9 serial interface, with the addition of +5V and auxiliary power.

Pin No.	I/O	Description/Comment
1	-	no connection
2	i/p	RxD - Receive Data
3	o/p	TxD - Transmit Data
4	o/p	DTR - Data Terminal Ready (input data stream handshake
5	-	Ground
6	-	no connection
7	o/p	+5Vdc power
8	i/p	CTS - Clear To Send (output data stream handshake)
9	o/p	Auxiliary power, eg 12V (if applicable)
case	-	Ground

1.2 Connector Details

An industry standard DB9 plug (male) is used for the connector.



Front View of DB9 Male Connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	10/04/95	First Issue	HR	

UMD Standard DB25 Parallel I/O Port

This document defines the UMD Standard DB25 socket Parallel Port.

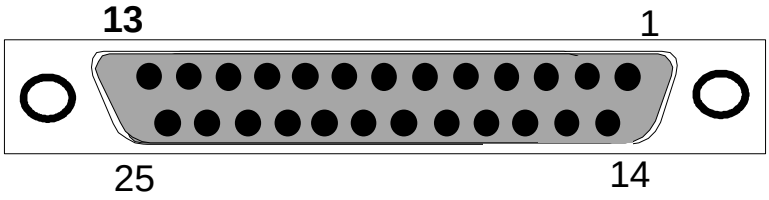
1.1 Pin Assignment

Pin assignments are a subset of the Centronics interface.

Pin	Description	As output	As input
1	Strobe(-)	O	I
2	Data 0	O	I
3	Data 1	O	I
4	Data 2	O	I
5	Data 3	O	I
6	Data 4	O	I
7	Data 5	O	I
8	Data 6	O	I
9	Data 7	O	I
10	Acknowledge(-)	I	O
11	Busy	I	O
12	Paper end	I	-
13	Select	I	-
14	Autofeed(-)	n/c	n/c
15	Error(-)	I	-
16	Initialise(-)	n/c	n/c
17	Select input(-)	Ground	Ground
18-25	Ground	Ground	Ground

1.2 Connector Details

An industry standard DB25 socket (female) is used for the connector.



Front View of DB25 socket Connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	06/10/95	First Issue	BMc	

UNIQUE MICRO DESIGN

Technical Standard 107

UMD Standard 7 Pin DIN Computer Keyboard and Power Interface

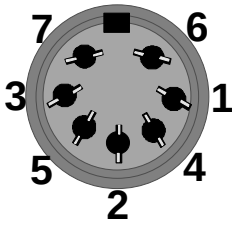
This document defines the UMD Standard 7 Pin DIN Computer keyboard and Power interface.

1.1 Pin Assignment

Pin No	I/O	Description
1	i/o	Clock
2	i/o	Data
3	-	Reset
4	-	Ground
5	i/p	5 Volt DC power
6	i/p	DC unregulated power in
7	o/p	5 Volt DC regulated out (jumper to pin 5, when using unregulated input on pin 6)

1.2 Connector Details

An industry standard 7 Pin Din connector is used.



Front View of 7 Pin Din socket Connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	10/10/95	First Issue	BMc	

UNIQUE MICRO DESIGN

Technical Standard 109

UMD Standard AMP DB9 BCR Wand Interface

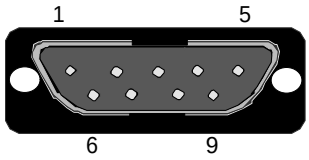
This document defines the UMD Standard AMP DB9 BCR Wand interface.

1.1 Pin Assignment

Pin	I/O	Description
1	-	no connection
2	i/p	data input
3	-	no connection
4	o/p	+5 volts DC
5	-	no connection
6	-	no connection
7	-	signal ground
8	-	shield ground
9	o/p	+5 volts DC

1.2 Connector Details

An industry standard AMP DB9 male (squeeze to release) connector is used.



Front View of DB9 plug squeeze to release connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	10/10/95	First Issue	BMc	

UMD Standard 5 Pin DIN Keyboard Interface

This document defines the UMD Standard 5 Pin DIN Keyboard interface.

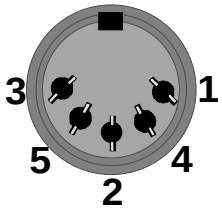
1.1 Pin Assignment

Pin assignments are as standard computer keyboard interface.

Pin	I/O	Description
1	i/o	Clock
2	i/o	Data
3	-	Reset
4	-	Ground
5	o/p	5 Volt DC

1.2 Connector Details

An industry standard 5 Pin DIN socket (female) is used.



Front View of 5 Pin DIN socket connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	11/10/95	First Issue	BMc	

UNIQUE MICRO DESIGN

Technical Standard 118

UMD Standard 5 Pin Din Cash Drawer Interface

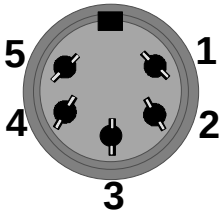
This document defines the UMD Standard 270° 5 Pin Din Cash Drawer interface for solenoid operated cash drawers with optional status feedback.

1.1 Pin Assignment

Pin	I/O	Description
1	o/p	coil positive
2	-	no connection
3	i/p	status feedback (return to ground)
4	-	no connection
5	o/p	coil negative
case	-	ground

1.2 Connector Details

An industry standard 270° 5 Pin Din socket (female) is used.



Front View of 270° 5 Pin Din socket connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	11/10/95	First Issue	BMc	

Standard DB15HD VGA Monitor Interface

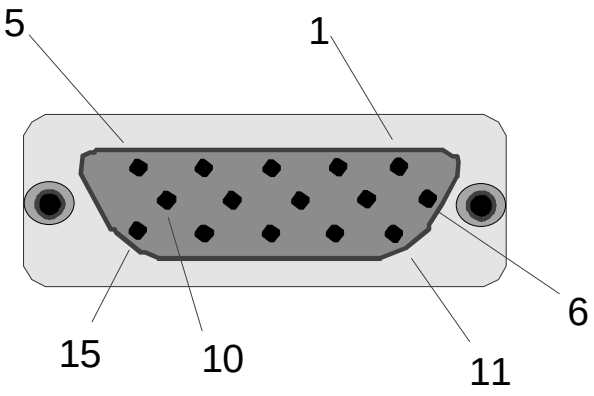
This document defines the Standard DB15HD socket, VGA Monitor interface.

1.1 Pin Assignment

Pin No.	I/O	Description/Comment
1	o/p	Red
2	o/p	Green
3	o/p	Blue
4	-	not connected
5	-	ground
6	-	ground
7	-	ground
8	-	ground
9	-	not connected
10	-	ground
11	-	not connected
12	-	not connected
13	o/p	Horizontal sync
14	o/p	Vertical sync
15	-	not connected

1.2 Connector Details

An industry standard high density DB15 socket (female) is used for the connector.



Front View of DB15HD Female Connector

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	20/10/95	First Issue	BMc	

UNIQUE MICRO DESIGN

Technical Standard 121

UMD ProtoLink Peripheral Interface

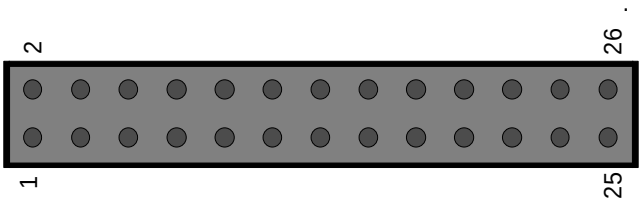
This document defines the UMD ProtoLink Peripheral Interface on a 26 way header.

1.1 Pin Assignment

Pin Number	Signal Name	Signal Direction	Function
1	**+5V**		**+5V Regulated Power**
2	**GND**		**Ground**
3	**A0**	**OP**	**Address bit 0**
4	**EXTRES**	**OP**	**External Reset active high**
5	**E1**	**OP**	**Enable 1 active high**
6	**R/W**	**OP**	**Read active high /Write active low**
7	**D1**	**I/O**	**Data bit 1**
8	**D0**	**I/O**	**Data bit 0**
9	**D3**	**I/O**	**Data bit 3**
10	**D2**	**I/O**	**Data bit 2**
11	**D5**	**I/O**	**Data bit 5**
12	**D4**	**I/O**	**Data bit 4**
13	**D7**	**I/O**	**Data bit 7**
14	**D6**	**I/O**	**Data bit 6**
15	**IRQ**	**IP**	**Interrupt request active low**
16	**A1**	**OP**	**Address bit 1**
17	**N/C**	**-**	**No Connection**
18	**E2**	**OP**	**Enable 2 active high**
19	**GND**		**Ground**
20	**+5V**		**+5V Regulated Power**
21	**A3**	**OP**	**Address bit 3**
22	**A2**	**OP**	**Address bit 2**
23	**A5**	**OP**	**Address bit 5**
24	**A4**	**OP**	**Address bit 4**
25	**GND**		**Ground**
26	**+5V**		**+5V Regulated Power**

1.2 Connector Details

An industry standard, 26 way, dual in line, 0.1 inch pitch male header is used.



Top View of 26 way header

---

2. Revision History

Issue	Date	Details	Author	Reviewed
1	10/1/96	First Issue	BMc	
2	06/03/96	Connector pin numbering corrected	BMc	