SCAN BARCODE (Single Scan)

This command is used to activate and read a barcode. For e280s, depending on the value of the trigger mode the scan laser may turn on (Soft mode) or require the user to press one of the scan buttons (Edge or Level mode). The encode value determines the data output composition. For M440 and M424 devices, since camera is used, hence scan buttons or laser is not applicable.

Note

- To implement multi-scanning, refer to BARCODE START (Multi-Scan) and BARCODE STOP (Multi-Scan) sections.
- SCA application has been enhanced to send CANCEL command (from Secondary Port) to cancel the scanning if the user requires to cancel.

Request Packet

Field	Rule	Type	Minimum	Maximum	Value(s)	Description
FUNCTION_TYPE	Required	Static value	N/A	N/A	BARCODE	Type of function
COMMAND	Required	Static value	N/A	N/A	SCAN	Command name
BCSCAN_TIMEOUT	Required	Numeric	500 (ms)			Scan barcode timeout, in milliseconds (ms). It is recommended to use 500 ms increments. **Example:** 5000
BCSCAN_ENCODE	Optionanl	List	N/A	N/A	0 – Encode both symbology and barcode data (default) 1 – Encode only barcode data (removes symbology data) 2 – Do not encode barcode data (removes symbology data)	Encodes barcode scan.

Field	Rule	Type	Minimum	Maximum	Value(s)	Description
BCSCAN_MESSAGE	Optional	Character		30		This field is sent to customize the message on UI display. This field is applicable, if BCCFG_DISPLAY_UI is set to 1 (enable) and then the message will be displayed on the UI screen. **Example:** `<BCSCAN_MESSAGE>SCAN HERE</BCSCAN>`. If this field is sent without any message (empty value), then the UI screen will display the default message as “Please Scan Barcode”. This is applicable to e280s, M440 and M424 devices.
COUNTER	Required	Numeric	1	10 (digits)		COUNTER is used for a given MAC label. Each COUNTER should be higher than the last one. This is sed to authenticate the POS. **Example:** 3
MAC	Required	Base64 Encoded Data	N/A	N/A		Message Authentication Code. This is used to authenticate the POS.

Field	Rule	Type	Minimum	Maximum	Value(s)	Description
MAC_LABEL	Required	Character	1	50 (digits)		Associated label that tells the device which MAC_KEY to use to decrypt the value of MAC. This is used to authenticate the POS. **Example:** P_JTY065

Example

Following is an example of request packet

```
<TRANSACTION>
<FUNCTION_TYPE>BARCODE</FUNCTION_TYPE>
<COMMAND>SCAN</COMMAND>
<BCSCAN_TIMEOUT>5000</BCSCAN_TIMEOUT>
<BCSCAN_ENCODE>2</BCSCAN_ENCODE>
<MAC>…<MAC>
<COUNTER>3</COUNTER>
<MAC_LABEL>P_JTY065</MAC_LABEL>
</TRANSACTION>
```

Response Packet

Field	Type	Value	Description
RESPONSE_TEXT	Character		Processor response text. **Example:** Command response 0
RESULT	Character		This indicates the Result details. **Example:** SUCCESS
RESULT_CODE	Numeric	Expected result code: -1, 59001, 59006, 59040	This indicates the result code. Refer to Result/Error Codes for details.

Field	Type	Value	Description
TERMINATION_STATUS	Character	SUCCESS or FAILURE	This indicates the transaction termination status. This is the overall status of the transaction irrespective of approved or declined. Like, if the output is generated then the status is SUCCESS and if no output is generated then the status will be FAILURE.
BARCODE_DATA	Character		Base64 encoded. Refer to the section below on Sample Code – Base64 Decoding for more details. **Example:** AAgEATY4OTU0NDA4MTY2MQ==
BARCODE_CODEID	Character		Returns unencoded. **Example:** 01
BARCODE_AIMID	Character		Returns unencoded. **Example:** 04
BARCODE_SYMBOLOGY	Character		Returns encoded. Returns only if BCSCAN_ENCODE = 0. **Example:** 0008
COUNTER	Numeric		Echoes counter sent in the request. **Example:** 3

Example

Following is an example of response packet

```
<RESPONSE>
<RESPONSE_TEXT>Command response 0</RESPONSE_TEXT>
<RESULT>SUCCESS</RESULT>
<RESULT_CODE>-1</RESULT_CODE>
<TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
<BARCODE_DATA>041415317635</BARCODE_DATA>
<BARCODE_CODEID>01</BARCODE_CODEID>
<BARCODE_AIMID>04</BARCODE_AIMID>
</RESPONSE>
```

Sample Code - Base64 Decoding

```
Base64 decode
inbuf: input buffer
outbuf: output buffer
insize: length of data in input buffer
outsize: length of output buffer (output data is approximately 25% smaller then input data, example a 40 byte
input buffer will result in a 30 byte output buffer)

static const char
cd64[]="|$$$}rstuvwxyz{$$$$$$$>?@ABCDEFGHIJKLMNOPQRSTUVW$$$$$$XYZ[\\]^_`abcdefghijklmnopq";

static void decodeblock(unsigned char in[4], unsigned char out[3])
{
out[ 0 ] = (unsigned char ) (in[0] << 2 | in[1] >> 4);
out[ 1 ] = (unsigned char ) (in[1] << 4 | in[2] >> 2);
out[ 2 ] = (unsigned char ) (((in[2] << 6) & 0xc0) | in[3]);
} // end decodeblock

void b64_decode(const unsigned char *pchInBuf, unsigned char *pchOutBuf, short shInSize, short *shOutSize)
{
unsigned char in[4], out[3], v;
int i;

short shLen;
short shIndex = 0;
short shOutdex = 0;

while(shIndex < shInSize)
{
for (shLen=0, i=0; ((i < 4) && (shIndex < shInSize)); i++)
{
v = 0;
while( (shIndex < shInSize) && (v == 0) )
{
v = pchInBuf[shIndex++];
v = (unsigned char) ((v < 43 || v > 122) ? 0 : cd64[ v - 43 ]);
if ( v )
{
v = (unsigned char) ((v == '$') ? 0 : v - 61);
}
} // end while
if (shIndex < shInSize)
{
shLen++;
if ( v )
{
in[ i ] = (unsigned char) (v - 1);
}
}
else
{
in[i] = 0;
}
} // end for

if (shLen)
{
decodeblock(in, out);
for(i = 0; i < shLen - 1; i++ )
{
pchOutBuf[shOutdex++] = out[i];
}
}
} // end while
*shOutSize = shOutdex;
```

```
} // end b64_decode
```

Code ID

Code ID	Description
01	UPC_EAN
02	CODE39_32
03	CODABAR
04	CODE128_ISBT
05	CODE93
06	INTL2OF5
07	DISC2OF5
08	CODE11
09	MSI
0A	GSI128
0B	BOOKLAND_EAN
0C	TRIOPTIC39
0D	COUPONCODE
0E	GS1DATABA
0F	MATRIX2OF5
10	UCCCOMPOS
11	CHINESE2OF5
12	KOREAN3OF5
13	PDF417_ISSNEAN
14	AZTEC_RUNE
15	DATA_MATRIX

Code ID	Description
16	QRCODE_MICRO
17	MAXICODE
18	US_POSTNET
19	US_PLANET
1A	JAPAN_POSTAL
1B	UK_POSTAL
1C	POSTBAR_CA
1D	NETH_KIX
1E	AUS_POST
1F	USPS_4CB
20	UPU_FICS
21	SCANLET_WEB
22	CUECAT

AIM ID

AIM ID	Description
01	CODE39_32
02	CODE128_ISBT_GS1
03	DATAMATRIX
04	UPC_EAN_COUPON
05	GS1DATABAR
06	CODEBAR
07	CODE93
08	CODE11
09	INTL2OF5

AIM ID	Description
0A	PDF417
0B	TLC39
0C	MSI
0D	QRCODE_MICROQR
0E	DISC2OF5
0F	MAXICODE
10	AZTEC_RUNE
11	X
12	COMP_EC
13	COMP_EE
14	COMP_RS

Symbology

Symbology	Description
0001	CODE39
0002	CODABAR
0003	CODE128
0004	D25
0005	IATA
0006	ITF
0007	CODE93
0008	UPCA
0009	UPCE
000A	EAN8
000B	EAN13

Symbology	Description
000C	CODE11
000D	MSI
000E	EAN128
000F	UPCE1
0010	PDF417
0011	CODE39FULL
0012	TRIOPTIC
0013	BOOKLAND
0014	COUPONCODE
0015	ISBT128
0016	MICROPDF
0017	DATAMATRIX
0018	QRCODE
0019	POSTNETUS
001A	PLANETUS
001B	CODE32
001C	ISBT128CONC
001D	POSTALJAPAN
001E	POSTALAUST
001F	POSTALDUTCH
0020	MAXICODE
0021	POSTBARCA
0022	POSTALUK
0023	MACROPDF417
0024	RSS14

Symbology	Description
0025	RSSLIMIT
0026	RSSEXPAND
0027	SCANLETWEB
0028	CUECAT
0029	UPCA_2
002A	UPCE_2
002B	EAN8_2
002C	EAN13_2
002D	UPCE1_2
002E	CCA_EAN128
002F	CCA_EAN13
0030	CCA_EAN8
0031	CCA_RSSEXPAND
0032	CCA_RSSLIIMIT
0033	CCA_RSS14
0034	CCA_UPCA
0035	CCA_UPCE
0036	CCC_EAN128
0037	TLC39
0038	CCB_EAN128
0039	CCB_EAN13
003A	CCB_EAN8
003B	CCB_RSSEXPAND
003C	CCB_RSSLIMIT
003D	CCB_RSS14

Symbology	Description
003E	CCB_UPCA
003F	CCB_UPCE
0040	KOR3OF5
0041	UPCA_5
0042	UPCE_5
0043	EAN8_5
0044	EAN13_5
0045	UPCE1_5
0046	MACROPDF