ABSTRACT

Since the introduction of SAS/QC® software into the SAS® System, SAS Institute has received numerous inquiries concerning the use of data collection devices with the SAS System. Because automatic data collection allows for rapid gathering of large amounts of data with reduced potential for human error, these devices have become an essential element of quality control programs for many industrial and manufacturing sites. Combining the data-gathering power of these devices with the power and flexibility of the SAS System provides an ideal combination for manufacturing sites interested in quality improvement.

To enable the SAS System to become a complete quality-improvement tool, SAS Institute is testing Release 6.03 of the SAS System for Personal Computers with several devices from Mitutoyo Inc., DataMyte Corporation, and Ohaus Scale Corporation, world-wide leaders in the production of data-collection equipment. The concepts discussed in this article are not limited to the data-collection equipment presently being tested. The techniques presented should be applicable to any data-collection device transmitting ASCII data via the PC RS-232 port.

DATA-COLLECTION DEVICES

A data-collection device is a general term used to describe any device that receives a signal sent from a measuring instrument, converts the signal into an ASCII data stream, and then is capable of transmitting the data stream to a computer via a serial port (RS-232). Data collection devices generally fall into two basic categories: batch processing and interactive processing devices. Batch processing devices interface with a host PC in a batch environment while the interactive processing devices interface with a host PC in a real-time mode. A general overview of the various devices that have been tested are given in the following sections.

Batch Devices

The DataMyte 762 is an example of a batch data-collection device. A measuring instrument (such as a micrometer) is connected to the DataMyte 762. Measurements can be collected via a measuring instrument or entered manually using the keypad on the 762. The 762 stores measurements in its own internal memory as they are recorded.

Once the communication link is set, a sequence of triggering commands is issued from the PC, instructing the 762 to transmit the data. The specific triggering commands are dependent on the DataMyte model that is being used to capture the data measurements.

Interactive Devices

The Mitutoyo Multiplexer MUX-10 is one example of an interactive data collection device. A maximum of three measuring instruments can be connected to the MUX-10, which possesses an RS-232 compatible interface. Once the communication link is set between the host PC and the multiplexer, the multiplexer transmits the data measurements to the host as soon as they are received from the measuring instruments.

The OHAUS® GT 400 Electronic Balance for weight measurements is another example of an interactive data-collection device. The GT 400 has an RS-232 bidirectional compatible interface. A simple triggering command is sent from the host to the GT 400 to initiate the communication link. Once the command is issued, data measurements are transmitted electronically through the RS-232 port. A variety of operating parameter settings are available with the GT 400. For example, the GT 400 can be configured to transmit data at timed intervals once the balance has stabilized.

THE ROLE OF THE SAS SYSTEM

The SAS System provides the communication link and data-capture interface between a host PC and both batch and interactive data-collection devices. The communication link is provided by the SASSIO driver, shipped with Release 6.03 of the SAS System for Personal Computers. The driver supports various protocols for data exchange, buffering, and higher baud rates than the DOS operating system supports. It can be used to set the communication parameters (such as, baud rate, parity, and so on) for the RS-232 port, in lieu of using the DOS MODE command.

Data capture is made possible by the versatility of the SAS DATA step. Once the communication link is established, the INFILE and INPUT statements in the DATA step provide the necessary tools for capturing the transmitted data into a SAS data set for data analysis and presentation.

With these capabilities, the SAS System is the data manager/presenter for data on any host system desired, such as, MVS, CMS, and VMS operating systems. Data transfer from the PC to the desired host is accomplished through the micro-to-host link or procedures in the base SAS product, such as the CPORT/CIMPORT procedures. The portability of the SAS System makes it the complete software interface for automated data collection, data management, and data analysis/presentation.

THE SASSIO DEVICE DRIVER

In order to read data via the RS-232 port, the SASSIO driver must be installed and the CONFIG.SYS file modified. The SASSIO driver provided with base SAS software (Release 6.03 of the SAS System for Personal Computers) enables the SAS DATA step to communicate with any data-collection device connected to the RS-232 port. To install the SASSIO driver, issue the following command from your $SASROOT:SASINST directory:

```
SASBACK -RESTORE A: C:SASSIO.SYS -SDIR -SET SAS=SASROOT -START n
```

where

- A: is the name of the source drive,
- C: is the name of the target drive,
- SASROOT is the directory in which SAS is installed.
- n is the diskette where the SASSIO.SYS driver resides. The values for n are the following:

498
Upon completion of the SASBACK command, the SASSIO driver will be installed into the 'SASROOT\SASMISC' subdirectory. To use the SASSIO driver, add the following line to your CONFIG.SYS file:

```
DEVICE='PATH\SASSIO.SYS' NAME='PORT' HARDWARE= 
BAUD= PARITY= DATABITS= STOPBITS= ASIS
```

where

- **PATH** specifies the drive and directory where SASSIO resides.
- **NAME** specifies the name of the device (i.e., COM1, COM2).
- **PORT** is 1 or 2.
- **HANDSHAKE** is NONE, XONXOFF, or HARDWARE.
- **BAUD** is 110, 150, 300, 600, 1200, 2400, 4800, 9600, or 19200.
- **PARITY** is NONE, ODD, EVEN, MARK, or SPACE.
- **DATABITS** is 7 or 8.
- **STOPBITS** is 1 or 2.
- **ASIS** indicates that the port’s BAUD, PARITY, DATABITS, and STOPBITS settings should be defined as they are set from DOS (using the DOS MODE for DOS versions 3.x or later).

The HANDSHAKE = option specifies the protocol of data exchange between the data-collection device and the host PC. The SASSIO driver supports three different handshaking methods:

- **NONE** specifies that there is no handshaking.
- **HARDWARE** specifies a flow control or pacing handshaking called CTS/RTS (Clear To Send/Request To Send).
- **XONXOFF** specifies a flow control method that uses the ASCII characters XON for transmit on and XOFF for transmit off.

For example, if the data-collection device is connected to the COM1 port and uses hardware handshaking and the communications parameters are set using the DOS MODE command, then the DEVICE= statement looks as follows:

```
DEVICE='SASSIO.SYS' NAME='COM1' PORT=1 HARDWARE=HARDWARE ASIS
```

Note that the ASIS option does not require any parameters. This option acts as a toggle, either ASIS is on or off. For some data-collection devices the SASSIO driver may not be required. However, as mentioned previously, the SASSIO driver provides support for higher baud rates, buffering, and XONXOFF protocol.

Once the SASSIO driver is installed and loaded (the CONFIG.SYS file modified and the PC rebooted), the SAS System treats the file modified and the PC rebooted), the SAS System treats the device as if it had been defined as a new device. The operation manual of the data-collection device should describe the format of the data being transmitted, as well as the communication parameters for the DOS MODE command, and the necessary triggering commands for initiating data transmission.

**INTERFACING THE SAS SYSTEM WITH THE MITUTOYO MUX-10**

A Mitutoyo Multiplexer MUX-10 is a data-collection device that receives measurement data from Mitutoyo Digimatic measuring instruments, modulates the data into an ASCII data stream, and then outputs them immediately to the attached personal computer. As stated previously, the MUX-10 can interface with a maximum of three measuring instruments. The MUX-10 uses HARDWARE handshaking and operates at baud rates of 300, 600, or 1200 bits per second.

Since the format of the data being transmitted is device dependent, the DATA step must be customized for the particular data-collection device. The operation manual for the MUX-10 states that the data stream is transmitted in fixed-length records of 13 bytes each, thus the LRECL= option of the INFILE statement is set to 13, and the RECFM= option is set to F. The SAS DATA step is designed to look ahead in the ASCII data file, searching for an end-of-file character (EOF or '1A'x). The UNBUFFERED option in the INFILE statement prevents the SAS System from look-ahead searching for an EOF character.

The first character transmitted from the MUX-10 is a code signaling whether data are being transmitted or whether an error has been encountered. If the code is 0, then data are being transmitted. If the code is 9, then a Mitutoyo error is being transmitted. If the value for code is not 0 or 9, then the data-collection device is malfunctioning and you should verify that it is set up properly. The second character transmitted from the MUX-10 identifies the channel from which the data are sent. The data value, including the sign, is in bytes 4 through 12.

Appendix A provides the DATA step program necessary to interface the SAS System with the Mitutoyo MUX-10.

**INTERFACING THE SAS SYSTEM WITH THE OHAUS GT 400**

An OHAUS GT 400 Electronic Balance is an interactive device capable of transmitting data to a host PC via a RS-232 port. In particular, the GT 400 uses HARDWARE handshaking and operates at baud rates ranging from 110 to 9600 bits per second. Several serial data frame settings are available, determining the number of data bits, stop bits, and the parity condition settings.

The triggering command necessary for data transmission is found in the operation manual for the GT 400. Specifically, the character F followed by a carriage return (CR), '0D'x, must be sent to the GT 400 to initiate data transfer. Once received, the GT 400 sends a data stream that is a maximum length of 20 characters or bytes, terminated by a carriage return/line feed. Since the SAS System recognizes a carriage return/line feed (CRLF) or just a line feed (LF) as a record terminator, it is not necessary to specify the LRECL= and RECFM= options in the INFILE statement. As discussed previously, the UNBUFFERED option prevents look-ahead reads by the DATA step.

The first byte transmitted is either blank or includes a minus sign indicating the sign of the data value. The next 8 bytes comprise the data value itself, right justified with leading blanks and a decimal point. The tenth byte is a blank followed by a 5-byte field that holds the measurement unit's abbreviation, such as GRAMS. The opera-
INTERFACING THE SAS SYSTEM WITH THE DATAMYTE 762

The DataMyte 762 is a batch data-collection device which has the capacity to store up to ten different characteristics about any part being measured. In addition to the measurement characteristics, the DataMyte 762 also stores information such as time, date, assignable cause, machine code, and operator code. The 762 uses XON/XOFF handshaking protocol and can operate at baud rates ranging from 300 to 9600 bits per second.

Once the data are stored within the 762, the device is connected to a PC via the RS-232 port. To initiate data transfer, the host PC must send a triggering command to the 762. The three basic triggering commands required to unload or capture the data are

?SETUP

?ITEMID/ITEM=n

?DATA/ITEM=n

The ?SETUP command prompts the DataMyte to return the number of items (or variables) being measured, the number of samples per subgroup, and the maximum number of subgroups stored. The information obtained from this command enables the SAS System to prepare appropriately for data collection and to construct a SAS data set necessary to perform further data analysis.

The ?ITEMID/ITEM=n command prompts the DataMyte for a label associated with variable n (where n ranges from 1 to 10). Since the 762 requires that the variable information be requested via numbers from 1 to 10, the acquisition of the variable label provides descriptive information important for accurate data analysis and presentation.

Lastly, the ?DATA/ITEM=n command requests that the DataMyte send the data for variable n. The DataMyte 762 sends data stored for a given variable or characteristic in a data stream called a packet. A packet follows the following format:

```
mm/dd/yy, bbb; ccc; data1; data2; data3 ... datal
```

where

- `mm/dd/yy` is the date and time.
- `bbb` is the machine code.
- `ccc` is the assignable cause code.
- `data1...datan` are the measurements per subgroup for item n.

Single or multiple packets may be sent by the 762. The last packet of information ends with an end of transmission or EOT ("04'x). Since the DataMyte 762 utilizes XON/XOFF handshaking, each data packet sent by the DataMyte includes a checksum value. If the checksum of the packet received by the PC matches the checksum sent by the 762, then the data have been successfully transmitted. Otherwise the PC sends the DataMyte a negative acknowledgement (NAK, or "15'x) and the data are retransmitted.

More details may be found in Appendix C, which provides the DATA step program necessary to Interface the SAS System with the DataMyte 762.

Applications Interface

Data acquisition with the SAS System provides the ideal platform for the development of site-specific, menu-driven application systems. Key to the development are the types of data-collection equipment used, the degree and complexity (simplicity) of the user interface desired, and the target system for detailed analysis. The base SAS System and SAS/QC Software residing on a factory floor, developed into a user-friendly interface, provides the operator with the ability to monitor data in a real-time fashion. The incorporation of the micro-to-host link opens the accessibility of the floor data to additional quality control analysts for use in other areas of the company. For example, the quality operator on the factory floor could be monitoring hourly data as in Figure 1, while another quality analyst is incorporating the hourly data into a daily chart as in Figure 2.

SUMMARY

The SASSIO driver provides a direct communications interface between data-collection devices and the SAS System via the RS-232 port of the personal computer. Once the data are captured into SAS data sets, all of the power of the SAS System is at the fingertips of the quality control analyst. Data from different manufacturing lines, different shifts, or different suppliers can be merged and analyzed. In addition, the data sets can be uploaded to a mainframe or minicomputer, using the SAS micro-to-host link, for further analysis by quality engineers, possibly at another site. With automated data collection and the power of the SAS System, the possibilities are endless.

ACKNOWLEDGMENTS

The authors wish to thank many individuals for their cooperation and support, including Bob Nadeau, NC Measurements Corporation; John Sims, SAS Institute Inc.; Larry Anderson, Mitutoyo Inc.; Jerry Houston, DataMyte Corporation; Dan Garland, Ohaus Corporation; SAS and SAS/QC are registered trademarks of SAS Institute Inc.; and DataMyte is a registered trademark of DataMyte Corporation.

OHAUS is a registered trademark of Ohaus Corporation.

APPENDIX A - SAS CODE FOR MUX-10

```sas
/*****************************/
/* Set up communications parameters */
/* (Band rate, parity, databits, stopbits) */
/*****************************/
set up communications parameters
x 'mode com1:1200,n,8,1';
/*****************************/
/* Set the total number of measurements */
/* to be taken from each channel */
/*****************************/
slet obschn1=10;
slet obschn2=10;
slet obschn3=10;
data mux10;
```
if _N_=1 then do;
  put / 'Ready to read data from the MUX-10.' / 'Press the LOAD or DATA button.' /;

/* Set up communications parameters */
/* (Baud rate, Parity, Databits, Stopbits) */
/* MUX-10 transmits 13 characters per measurement. */
totobs=t; obschn1+tobschn2+tobschn3;
end;

/* Read 5 measurements from the OHAUS GT 400. */
infile 'COM1:' recfm=f lrecl .. 13 unbuffered;
do obs=1 to totobs;
  input code 1 01;
/* If code .. O then data are being transmitted. */
  if (code=0) then do;
    input channel 2 value 4-12;
    put 'Channel= channel ' Measurement= ' value output;
  end;
/* Error Handling - If Code=9 then an error has occurred. Read in error code and channel number. */
else if (code=9) then do;
  input channel 2 error 3;
  if error .. l then do;
    put / 'ERROR: NO DATA ENTRY DETECTED FROM CHANNEL'
      'channel=' channel=' error= ' /
      output;
  end;
  else if error=2 then do;
    put / 'ERROR: UNRECOGNIZED DATA FORMAT FROM CHANNEL'
      'channel=' channel=' error= ' /
      output;
  end;
  else do;
    put / 'ERROR: UNRECOGNIZED ERROR DETECTED';
    put / 'channel=' channel=' error= ' /
    output;
  end;
stop;
end;
end;
run;

APPENDIX B - SAS CODE FOR OHAUS GT 400

 APPENDIX C - SAS CODE FOR DATAMYTE 762

APPENDIX B - SAS CODE FOR OHAUS GT 400

/* Set up communications parameters */
/* (Baud rate, Parity, Databits, Stopbits) */
x 'node com1=9600,n,8,1';
filename in 'COM1:';
filename out 'COM1:';
data ohaus;
  file out;
infile in unbuffered;

/* Read 5 measurements from the OHAUS GT 400. */

APPENDIX C - SAS CODE FOR DATAMYTE 762

/* Set up communications parameters */
/* (Baud rate, Parity, Databits, Stopbits) */
x 'node com1=9600,n,8,1';
filename in 'com1:';
filename out 'com1:';
data temp;
retain xoff='02'; keep '04'; xoff '13'; xon '11'; packet = 1;
length prestr $ 3 firststr $ 200 calccs $ ccs $ 2;
length sec $ 200 command $ 25;
file out lrec=r recfm=f;
infile in lrec=r recfm=f unbuffered;

/* Issue the ?SETUP command to query the device for */
/* the number of items and number of samples per */
/* subgroup. */

cmd='?SETUP';
link SENDCKD;
link READIT;

do item = begin to tot;
   id=item;
   subgroup=9;
   packet = 1;
   /**************
   /* Issue the ?ITEM/ITEM- command to receive */
   /* descriptive info (labels) for each item. */
   *******/
   cmd='?ITEM/ITEM-': illeft(item);
   link SENDCKD;
   link READIT;

   /**************
   /* Issue the ?DATA/ITEM+ command to receive the data */
   *******/
   cmd='?DATA/ITEM+': illeft(item);
   link SENDCKD;
   link READIT;

   end;
   /* do item end */
   step;
return;

   /**************
   /* Subroutine code for READIT */
   /* links to subroutines REVCNT, CHKSUK, ACKNAK */
   *******/
   READIT:
   prestr='';
   str='';
   firststr='';
   cs='08':
   calcset='06':
   calccs='00':
   xofflag='-1;
   input x $char1;

   /**************
   /* Read packets until an EOF is encountered */
   /* if x is a STX then read in 2 byte character count */
   *******/
do while( x ne EOF );
   firststr='';
   str='';
if x eq STX then do;
   input i $char1;
   input i $char1;
   prestr='';
   command='';
   link REVCNT;
   end;
   if x eq stx end /*
   reread1;
   flag='';
   index=0;

   /**************
   /* The character count tells us how many characters are in the */
   /* packet being sent from the data collector (datacnt is number */
   /* of chars calculated in revcnt). */
   *******/
do i = 1 to datacnt;
   index=0;
   input x Scharl.;
   substr (str, index, 1 ) = x;
   if (Ix eq ';'1 and (index gt 175)) then do;
      put XOFF; /* tell the datamyte xoff to wait till we process */
      what we have already received */
      STR=prestr: illeft(str);
   end;
   end;
   put XON;
   xofflag=1;
end;
   /* if x eq stx end */

   /**************
   /* Set XON to tell the 762 to continue. */
   *******/
pot XON;
   xofflag=1;
end;
   /* ends large string do loop */
   /* this ends dataset loop */

   /**************
   /* Input checksum values from the 762. */
   *******/
   input cksum1 $char1. cksum2 $char1.;
dcksum=cksum1:cksum2;
   if xofflag=1 then str=prestr=illeft(str);
link CHKSUM;

link ACKNAK;

link COUNT;

if x eq -1 then do;
    firststr=substr(firststr,1);
    prefllaq=index(str, 6) - 6;
link COUNT;
end;

if last character in string is not EDT then multiple packets
are sent

if substr(cmd,1,2)='?D' then packet+1;

xofflag=-1;
/* end of DO WHILE loop */

return;

CHARCNT:

if flag eq '06's then do;
    firststr=substr(firststr,4);
    prefllaq=index(str, 8) - 8;
link COUNT;
end;

XOFFLAG:

if xofflag eq -1 then do;
    cs='00'x;
    firstcs='00'z;
end;
else cs .. firstcs;

len=length(str);
do index,1 to len;
    bit=7;
    rbval=rank(cs 1);
    rsval=rank(substr(str, index, 1));
    sum=0;
do until(bit=0);
    if(int(rbval=2*bit) y= int(rsval/2*bit))
    then sum=sum+2**bit;
    if(rbval=2*bit) then rbval=rbval-2*bit;
    if(rsval=2*bit) then rsval=rsval-2*bit;
    bit=bit-1;
end;

cs=byte(sum);
end; /* end of do index */

CHKCNT:

link ACKNAK;

if flag eq '06's then do;
    firststr='00's;
end;
else cs=firststr;

len=length(str);
do index=1 to len;
    bit=7;
    rbval=rank(cs 1);
    rsval=rank(substr(str, index, 1));
    sum=0;
do until(bit=0);
    if(int(rbval=2*bit) y= int(rsval/2*bit))
    then sum=sum+2**bit;
    if(rbval=2*bit) then rbval=rbval-2*bit;
    if(rsval=2*bit) then rsval=rsval-2*bit;
    bit=bit-1;
end;

cs=byte(sum);
end;
/* end of do index */

ACKNAK:

if the 762 checksum and the calculated checksum match then
/* send an ACK, '06's */
endif checksum=calccs then do;
    flag='06's;
    put flag $char1.;
end;

endif checksum=calccs then do;
    flag='06's;
    put flag $char1.;
    link READY;
end;
else do;
    file out;
    put '02'b;x;
    FILE LOG;
    PUT / "ERROR: Reading data retry exceeded" /;
    PUT _ALL_;
    STOP;
    END;
    return;

/* End subroutine to calculate the character count. */

/* This subroutine sends the DataMyte commands to the device */
/* and checks for a response. If ENQ received then a */
/* invalid command has been entered. If ACK is received */
/* then command has been accepted. If a NA is received */
/* then the command is resent to the device until the */
/* retry count of 5 is exceeded. */
/* valid command has been entered. */
/* processing and issue an ABORT (STX,ESC) to the device. */
/* else exit */
/* send an abort (STX,ESC) to the device and print a */
/* message to the log saying the reading retry was exceeded, */
/* and processing halted. */
/* /** If a NA is still encountered after 5 retries, then halt */
/* **/ If an ENQ is received, check the resend counter, and if le 5, */
/* **/* reissue the command to the device. */
/* **/ return if (x eq '05'x) then link sendcmd;
/* **/ PUT _ALL_;
/* **/ Link to routine to calculate the checksum. */
/* **/ Link to routine to read in a character. */
/* **/ Send the command to the device */
/* **/ and read in a character to see if an ACK is there. */
/* **/ If the checksum is the exclusive or (XOR) of all the characters */
/* **/ in the command including the STX, character count (CC), */
/* **/ and EOT. */
/* **/ Count the number of words in the packet. */
/* **/ This determines the number of words. */
/* **/ and splits the words into date/time, operator, machine, */
/* **/ acronym, and process variables. */
/* **/ COUNT:
/* **/ /* /** Count the number of words in firststr (less than 175 chars). */
/* **/ /* /** Count the number of words in str. */
/* **/ str=firststr:es:command;
/* **/ /* /** Count the number of words in firststr (less than 175 chars). */
/* **/ end;
/* **/ /* /** Count the number of words in str. */
/* **/ y1="";
/* **/ ocount=0;
/* **/ Count the number of words in firststr (less than 175 chars). */
/* **/ end;
/* **/ /* /** Count the number of words in str. */
/* **/ if c ne 0 then count+=cntstr;
/* **/ Read the command line count Enters. */
/* **/ count=sum(cntstr,cntfirst) ;
/* **/ Count the number of words in firststr (less than 175 chars). */
/* **/ q=0;
/* **/ Read the command line count Enters. */
/* **/ q+=cntstr;cntfirst;
/* **/ Count the number of words in firststr (less than 175 chars). */
/* **/ end;
/* **/ /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
/* **/ Scan count characters from firststr. */
/* **/ word=scan(firststr,w,str);
/* **/ /* /** Separate the string into words based on semicolons */
/* **/ word=scan(firststr,w,str);
/* **/ end; /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
/* **/ Scan count characters from firststr. */
/* **/ word=scan(firststr,w,str);
/* **/ end; /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
/* **/ Scan count characters from firststr. */
/* **/ word=scan(firststr,w,str);
/* **/ end; /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
/* **/ Scan count characters from firststr. */
/* **/ word=scan(firststr,w,str);
/* **/ end; /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
/* **/ Scan count characters from firststr. */
/* **/ word=scan(firststr,w,str);
/* **/ end; /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
/* **/ Scan count characters from firststr. */
/* **/ word=scan(firststr,w,str);
/* **/ end; /* /** Count the number of words in str. */
/* **/ do w=count to (count-1);
*/
begin;
call symput('beg',left(begin));
end;
when (2)
    samp=word;
otherwise return;
end;
when (1)
    label=substr(word,1,(x-1));
end;
when (1)
    call symput('beg',left(begin));
end;
otherwise return;
end;
when (2)
    samp=word;
otherwise return;
end;
when (1)
    label=substr(word,1,(x-1));
end;
otherwise do;
    mod=word(w,(#=samp));
select (mod);
when (0)
    date=word;
    subgrp=word;
sample=sample;
end;
when (1)
    operator=word;
when (2)
    machine=word;
when (3)
    cause=word;
otherwise do;
    process=word;
sample=sample;
if z=0 then do;
    output;
end;
end;
otherwise do;
    process=substr(process,1,(z-1));
    output;
end;
return;
run;
proc sort data=temn; by subgrp sample;
run;
proc print; run;
/ preliminary remarks;
Figure 1  Hourly Data Monitored on Factory Floor

Figure 2  Weekly Data Monitored by Quality Analyst