

Design Trends

A quarterly publication brought to you by Motion Designs Inc.

August 2008

In this issue of Design Trends:

- Technology: What you need to know about CANpage 1
- New Product: Arcus Nema 11 K-SA integrated stepperpage 6
- Product Feature: Technosoft Binary Code Viewer.....page 7
- Application Solution: Stegmann all digital absolute motor feedback.....page 10

What you need to know about CAN

CAN (Controller Area Network) is a high-speed serial bus system for embedded networking and control. More than 2 billion CAN nodes have been sold since its inception in the early 80's. CAN received international standardization in 1993 (ISO 11898-1).

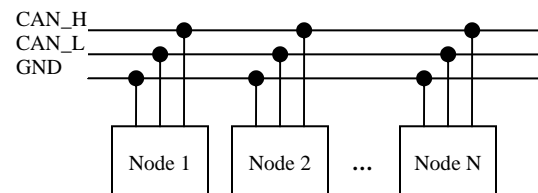
As yet another industrial networking "battle" is taking place - with Ethernet at the center stage - it is worthwhile revisiting how CAN works and also compare it with the oldest (RS232/485) and newest (Ethernet) networking technologies out there.

It must be clearly recognized that there are 2 main aspects to any communication system. The first aspect is the physical aspect (wiring, electrical signals, connectors etc...). The second main aspect is the protocol (i.e. the data packages and their meaning). A communication network is not

completely defined by just one of these 2 components. This is especially true in reference to Ethernet which ONLY defines the physical aspect NOT the actual data.

CAN Physical Wiring

A CAN bus is a 3-wire differential bus consisting of a CAN_H, CAN_L, and GND wire. Multiple devices are connected to a common bus in a direct fashion:



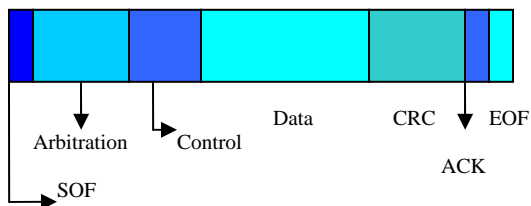
Nominal voltage levels are 3.5V and 1.5V for CAN_H and CAN_L respectively in a dominant state

(considered logic 0) and 2.5V for both in a recessive state (considered logic 1). Maximum bit rate is 1Mbit/s up to a maximum 25 m line length. Bit rates of 500 Kbits/s and below can be used for line lengths L (in meters) abiding the following rule:

$$BR [Mbit/s] * L [m] \leq 60.$$

To avoid transmission line effects, a 120 Ohm resistor must be placed between CAN_H and CAN_L on both ends of the main line.

Communication over the CAN network is done via so-called CAN frames. Presently there are 2 standards, CAN2.0A and CAN2.0B, with the following CAN frame definition:



- SOF: Start Of Frame is a single dominant bit.
- Arbitration Field: in the case of CAN2.0A this fields consists of an 11 bit identifier plus 1 RTR bit. In the case of CAN2.0B this field consists of an 11 bit base identifier, 1 IDE bit, 1 SSR bit, 18 bit extended identifier and 1 RTR bit.
- Control Field: 6 bits that determine message type (2.0A or B) and data length.
- Data Field: 0 to 8 data bytes (i.e. 0 to 64 bits).
- CRC Field: consists of a 15 bit CRC sequence and 1 delimiter bit.
- ACK Field: consists of a 2 bit acknowledgment sequence

- EOF: End of Frame is a 7 bit sequence.

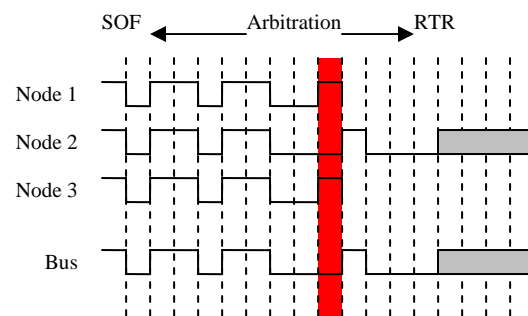
Each frame is separated by an intermission frame of 3 bits.

In addition, because CAN utilizes NRZ encoding, there is a bit stuffing mechanism that adds a bit of opposite polarity after 5 equal bits in a bit stream.

So in summary, a CAN message frame can be up to 130 (150) bits for CAN2.0A (CAN2.0B respectively).

Because the basic topology is a bus structure with a common medium, an arbitration mechanism must resolve bus access. CAN utilizes CSMA/CD (Carrier Sense Multiple Access Collision Detection) with non-destructive resolution. It is this last feature that makes CAN a very good candidate for real time deterministic networking.

The following picture demonstrates what happens when multiple nodes attempt to transmit a message at the same time:



All nodes start with their SOF bit, and then proceed with their arbitration field. The first 7 arbitration bits are identical for all 3 nodes. However, node 2 asserts a dominant bit at the 8th bit, whereas node 1 and 3 leave the bus recessive. Node 1 and 3 recognize that the bus is in a dominant state and stop

transmission of their bit stream. Node 2 simply continues transmission. Hence none of the data transmitted up to the point of arbitration is lost, resulting in non-destructive collision detection and resolution.

It is clear that the (properly named) arbitration field determines a clear message priority.

Beyond the CAN physical layer description provided here, there are a few very well defined protocols that capture specific functionality:

- **CANopen:** an internationally standardized protocol for embedded control systems, it includes a communication profile as well as application, device and interface profiles.
- **DeviceNet:** an internationally standardized protocol for industrial automation.
- **J1939-based:** an SAE1939 based protocol mainly used for in-vehicle networks.

Of course it is possible to define ones own protocol as well.

Comparison with RS232/422/485

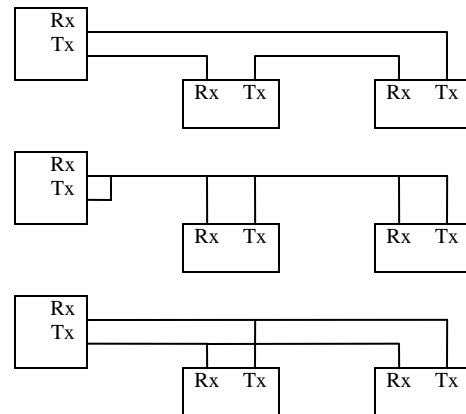
The RS's are one of the oldest and most universal serial network technologies in existence. Despite its general and broad acceptance, the RS stands for Recommended Standard and there is no official hard standard (such as ISO or IEEE).

RS422 and RS485 are basically differential versions of the single ended RS232. Although there is a handshake mechanism via additional wires, the

most common implementation uses a dedicated receive and transmit line (or differential pair) and a ground reference.

Voltage levels vary widely; $\pm 5V$, $\pm 10V$, $\pm 12V$ and $\pm 15V$ are all possible.

Various topologies are possible, such as daisy chain, half or full duplex.



Typical (although not official) baud rates are 9600, 19200, 38400, 57600 and 115200 bit/s (as found on most PC's). The differential RS422 and 485 can be practically used up to 10Mbit/s, but this is rarely done in communication systems.

Technically speaking there is no formal definition of the actual data packages, but an 8-bit (1 byte) data package with a single start bit, 1 or 2 stop bits and a parity check is the most common.

Compared to CAN, the RS interfaces are much less formal. In addition to the physical wiring and electrical signals, the following main differences stand out:

- **Data package:** typical RS data packages are 8 bytes, while CAN has a variable data size, up to 8 bytes.
- **Error checking:** other than a parity check on individual data packages,

any other error checking needs to be implemented in software or firmware. The user is responsible for re-transmitting a failed data message. CAN frames include a 15-bit CRC which is generated by the CAN controller silicon. There is also an automatic re-transmit mechanism, so no firmware or software has to be developed for this aspect of the communication system.

- Determinism: there is no formal collision detection and arbitration mechanism in RS networking (not applicable for daisy chained systems since they avoid collisions altogether). CAN has a well defined arbitration mechanism that makes it useable in real time applications.

Since many 8, 16 and 32-bit microprocessors and DSP's are now available with a built-in CAN interface, CAN is a very valid and cost-effective alternative to the traditional RS interface.

Comparison with Ethernet

Ethernet is one of the best defined, yet least understood networking technologies in existence today. Since it is transparently used in computer and office networking it has captured large attention for use in industrial applications.

Technically speaking Ethernet is standardized as IEEE 802.3, a collection of standards that define the physical layer and media access of wired Ethernet. This started as a 2.94 Mbit/s coaxial bus in 1972, but today covers copper, fiber-optic, and coax, with speeds ranging from 10Mbit/s to 40Gbit/s (this last one as a study group).

It is the 100Base-TX (IEEE802.3u) known as Fast Ethernet that most everybody refers to when speaking of Ethernet. With this comes to mind the 8-position RJ45 connector and CAT5 cabling.

The electrical signaling with Ethernet is much more complex than RS or CAN. First of all, any data to be transmitted goes through so-called 4B/5B coding. Every group of 4 bits is mapped into a 5 bit pattern. The reason for this is to provide transitions that allow multiple nodes in the network to stay synchronized (keep in mind that the bit time of a 100Mbit/s link is a mere 10 nanoseconds long!!!). This mapping is done with a pre-determined table.

After this 4B/5B translation, the bit stream is encoded using MLT-3 (Multi-Level Transmit) which uses 3 voltage levels. The bus voltage cycles through the 3 consecutive voltage levels when there is a bit level change in the data stream. This has the advantage of reducing the effective bandwidth requirement of the physical medium. Specifically, starting with 100Mbit/s, after the 4B/5B encoding, the effective bit rate is 125Mbit/s. However the fundamental frequency of the MLT-3 coding is one fourth of the bit rate, hence the physical layer bandwidth requirement is reduced to 31.25 MHz.

Ethernet frames (per the IEEE802.3 standard) consist of:

- A 56 bit preamble
- An 8 bit SFD (start of frame delimiter)
- A 48 bit destination address
- A 48 bit source address
- A 16 bit Length field
- Data field 46 bytes to 1500 bytes

- A 32 bit Frame Check Sequence (CRC)

Note that there is no automatic re-transmit in case the CRC fails. That is left to a higher layer protocol.

So a minimal Ethernet “datagram” is 72 bytes, or 576 bits.

Although in principal Ethernet (like CAN) uses a CSMA/CD mechanism, in general collisions are avoided all together. This becomes clear when one looks at what is actually wired inside the RJ45 and CAT5 cables. There are 2 differential pairs for transmit (TX+ and TX-) and receive (RX+ and RX-). Nodes are connected to each other or to a switch, hub or router by connecting the TX to the RX port and vice versa. Hence the TX and RX pairs are never tied together. Depending on the implementation of a node, despite the full separation of transmit and receive lines, it may still not be possible to run in full duplex (certainly true for a hub where 2 data frames may need to go out to the same port). The collision detection and resolution are quite different from CAN. When a collision is detected, the data transmitted up that point is discarded, and every node that participated in the collision backs off from re-transmission for a random time.

As mentioned before, collisions can be avoided by the use of intelligent switches, but this will also affect the timing of messages on the network.

These are the reasons why standard Ethernet is not deterministic (from a timing perspective).

There are a variety of protocols one can choose from that run on Ethernet, some of which leave the standard Ethernet intact but remain non-deterministic, some of which modify the underlying Ethernet layer to varying degrees to achieve some level of determinism. The most frequently encountered are:

- **Ethernet/IP**: this is DeviceNet on Ethernet.
- **Profinet**: this is Profibus on Ethernet.
- **EtherCAT**: re-uses the CANopen application protocol and achieves high-speed determinism by modifying the Ethernet interface on the slave nodes.
- **PowerLink**: also re-uses CANopen, achieves determinism at a slower cycle rate, but leaves the Ethernet network intact.

There is no question that the Ethernet physical layer has higher raw bit rates, but at a much higher cost and complexity. These higher bit rates can certainly compensate for lack of determinism in certain applications. The choice of protocols is very broad, although the “visibility” between application data and what goes “over the wire” is mostly lost.

In conclusion, a network architecture choice should not be based on pure speed. Data throughput on a network can be effectively reduced by proper distribution of the overall system control. Unless it is necessary to duplicate a high speed centralized control topology (e.g. closing a feedback loop) a slower network with intelligent nodes can be more cost effective and easier to implement.

DMX-K-SA-11 Integrated Nema 11 Motor/drive/controller from Arcus

- RS485 communication, multi-drop capable
- Operates in stand-alone or online mode
- Integrated encoder
- 24V Opto-isolated limit and home inputs
- 24V Opto-isolated output
- Absolute or relative position moves
- Various homing routines
- Up to 16 micro-steps per step
- 12 - 24 VDC supply
- Configurable current from 100mA to 1.5A
- Various stack sizes



Complete configuration and programming with Arcus setup software:

DMX-K-SA-11 v311

Status
 Position: 8000 R
 Encoder: 2560 R
 Status: IDLE C
 Current: 500
 Mode: ABS
 +L ○ H ○ -L ○

Communication
 Communication OK
 Baud Rate: 9600
 Device ID: 01

Program Control
 Status: Paused Index: 5
 Run Stop Pause Cont
 Text Program Total: 10

```

WHILE 1 = 1
X8000
WAITX
X0
WAITX
ENDWHILE
END
    
```

 Clear Code Space

Control
 Position: 1000
 High Speed: 1000
 Accel: 100
 Enable:
 Set Pos: [Home] [JOG+] [JOG-] [ISTOP] [ABS]
 Set Enc: [L+] [L-] [H+] [H-] [INC]

Product Information
 Firmware: V317
 Product ID: DriveMax-K-SA
 New Open

Digital Output
 D01

Variables Setup Compile Download Upload View Exit

Product Feature: Technosoft Binary Code Viewer

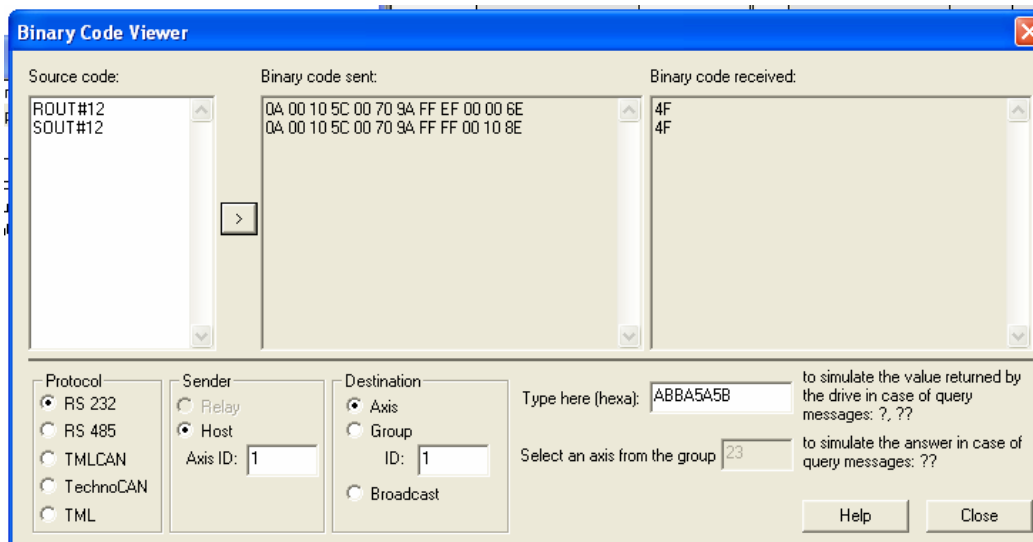
Embedded networking in industrial systems has become general, wide spread practice for quite some time. Despite the existence of both high speed and open standard communication technologies, cost and time-to-market may not allow use of the ideal solution. Also, some open standards do not allow application specific features, which may eliminate their use.

In these cases simple RS232, RS485 and/or CAN based technologies may be both sufficient and more cost-effective to do the job. Although certain communication protocols exist that operate with these low cost serial networks, they may create too much overhead and not enough flexibility. So very often, the communication protocol that comes with the device is utilized because:

- it exists (many open standard protocols only exist on paper or are in the process of being materialized)
- allows full flexibility
- is proven
- is typically well defined
- is typically very efficient

The largest downside is the learning curve of a “new” protocol. These manufacturer specific protocols are often accompanied with 200 page manuals, and if the protocol is worth its name, can also include a sizeable error checking mechanism. None of these attributes lend themselves to “fast deployment”.

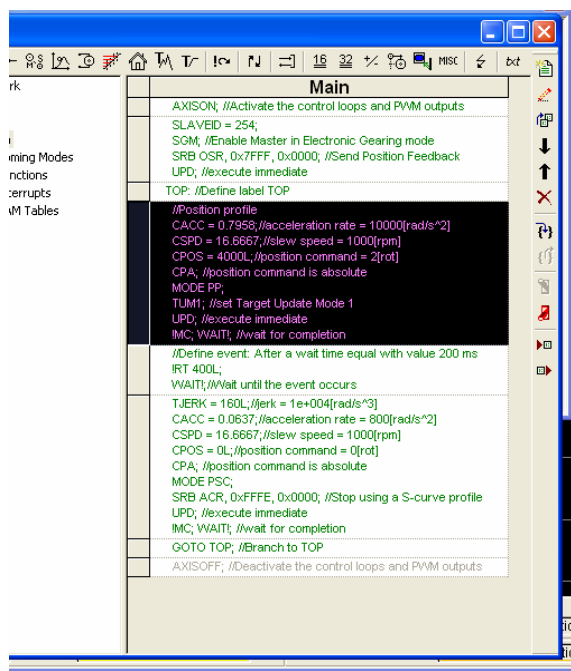
In order to accommodate the use of their communication protocol, Technosoft has included a so-called “Binary Code Viewer” into their EasyMotion Studio application software:



Technosoft drives are programmed in TML (Technosoft Motion Language), a

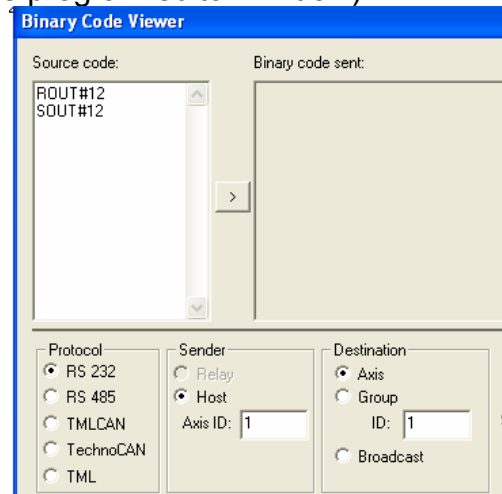
very powerful and compact higher level programming language that provides

complete access to the drive and allows creation of very complex and highly distributed motion and machine control architectures. Although complete programs can be developed without actually “writing” code (thanks to a very powerful editor which creates code automatically based on higher level function blocks, see below), it may be necessary to send TML commands from a PC or any other embedded device (host CPU, HMI, other drive...).



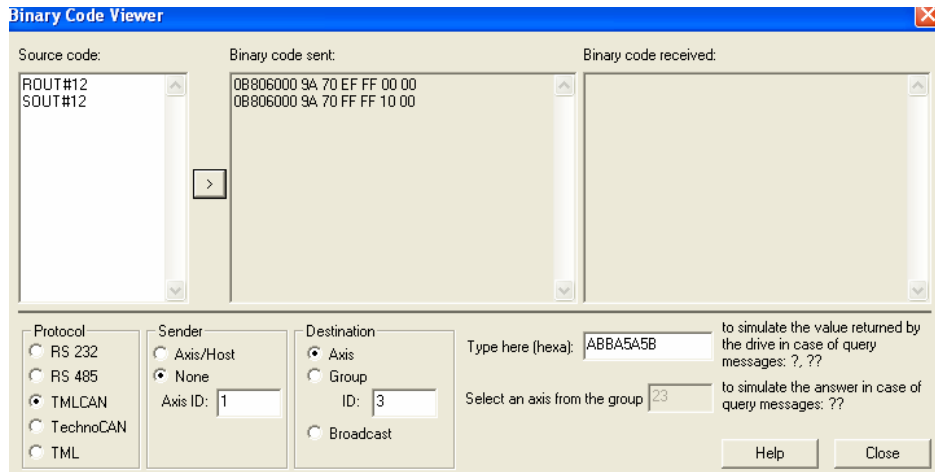
Technosoft drives support several communication channels including RS232, RS485 and CAN. In addition, the CAN interface supports various protocols: TMLCAN is the basic RS232/485 protocol over CAN, the CANopen open standard protocol and TechnoCAN, an extension of CANopen that allows access to all drive features through standard TML.

The Binary Code Viewer allows the user to quickly assemble any TML command over any interface with any protocol (except CANopen which of course follows the open standard). As an example, below are two TML commands, one for setting and the other for clearing an output (these commands can be easily copied and pasted from the program editor window):



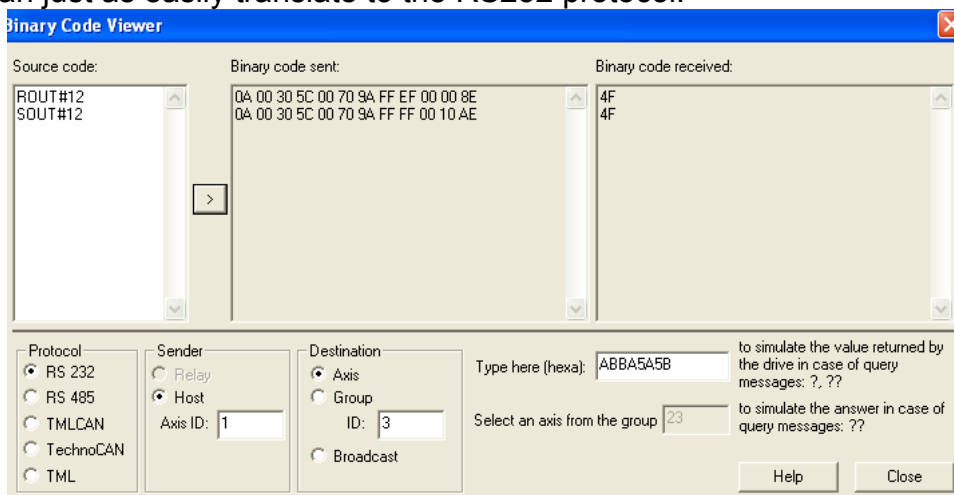
In addition to selecting the particular protocol that the commands need to be translated to, the user can also select the sender and destination parameters. That is due to Technosoft drives' capability to be network-able and hence addressable. This allows drive inter-communication but of course also allows communication with a non-Technosoft drive.

In the example, we translate the 2 TML commands that affect an output as commands to be sent from a host with ID 1 to a drive with ID 3, over a CAN link running the TMLCAN protocol:



The binary (more precisely hexadecimal code) to be sent over CAN is shown with both the CAN identifier and data fields.

But one can just as easily translate to the RS232 protocol:



In this case, not only the byte packages that will go over the serial link are shown (again as hex-values), but also the response from the receiver, since in this case there is an acknowledgment message. Furthermore, specific replies can be simulated, and the user can enter a data pattern that is to be used in the reply (and that can be easily distinguished from the other bytes).

The Binary Code Viewer feature makes communication development both faster and more accurate. And if the user is so inclined to learn the details of the various protocols, it also provides a handy tutorial to verify accuracy of manually translated commands.

Application Solution: Stegmann all digital absolute motor feedback

A manufacturer of metal cutting systems had been using conventional incremental encoder technology used as feedback on the servo motors. In addition to the incremental encoder feedback, Hall sensors are used for brushless commutation.

The incremental encoder resolution utilized is 8192 lines per revolution, resulting in 32,768 counts per motor revolution. The combination of incremental and Hall sensor signals results in the following wire count:

- 2 wires for power and ground
- 6 wires for differential A, B and I
- 6 wires for differential U, V and W

Although the wire count can be reduced by use of single ended signals, the high speed operation (at 3,000 rpm the encoder line frequency is 409.6 kHz) and industrial environment would make a non-differential interface very error prone.

The system builder wanted to increase resolution by a factor of 4 and reach similar high speed operation. In addition, it was desirable to reduce the wire count for the feedback system.

Of the many different approaches, one very attractive solution is an absolute encoder with serial interface. The absolute nature would eliminate the need for Hall sensors used for brushless commutation. The serial interface would reduce the wire count.

One particular encoder of this nature is a Stegmann single turn absolute encoder with SSI (Synchronous Serial Interface). This encoder has the following features:

- Differential clocked serial interface
- 16, 17 or 18 bit resolution
- Gray scale metal encoder disc
- Simple mono-flop shift register interface

Although many serial encoder interfaces exist today, they require very high clock rates and require complex circuitry and algorithms to read the position.

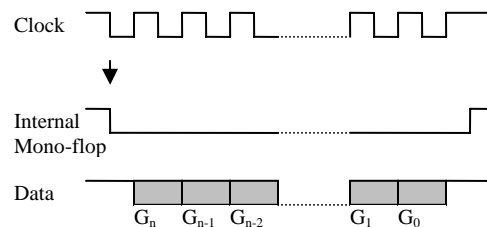


The Stegmann encoder is a text book example of elegance in simplicity.

The SSI interface consists of 6 lines:

- Power and ground (V and GND)
- Differential clock (Clock+ and Clock-)
- Differential data (Data+ and Data-)

Reading position is simply done by “clocking” out the position bit stream:

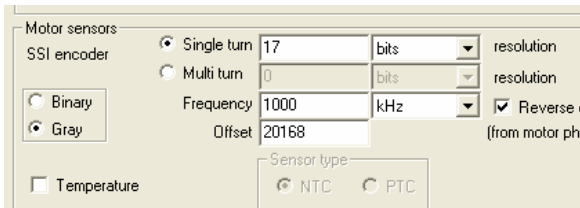


Upon a first falling edge of the clock, an internal mono-flop captures the position.

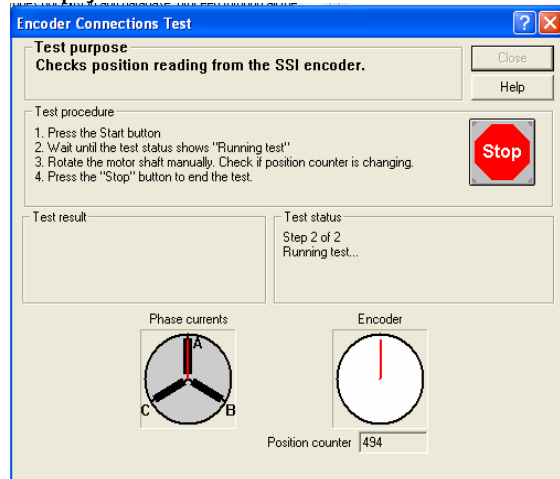
This ensures that the position is locked-in; this is of extremely high importance in data-sampled systems. Then, upon each rising edge of the clock, the position bits, starting with the most significant bit of the Gray pattern, are read up. Again, it is important to note that via the clock signal the user determines the sampling time. If very high performance is required, this feature allows for extremely precise position control (e.g. one can compensate for the time required to read the position).

An actual realization was done with a Technosoft drive, which supports the Stegmann SSI interface. Below is an example of a system running with:

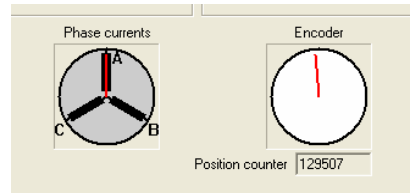
- A modest 1 MHz clock rate
- 17-bit single turn absolute position



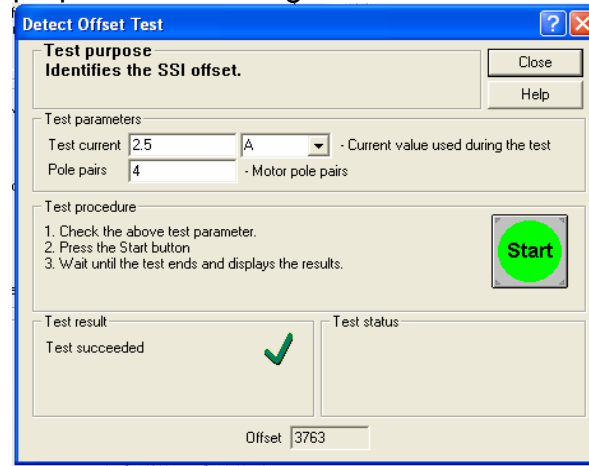
The actual interface can be tested very simply as follows:



After one motor revolution, the position rolls over after 131,071 counts ($2^{17} - 1$).

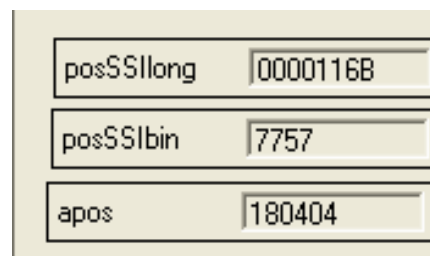


For commutation purposes, an automatic offset detection ensures proper electrical angle calculation:



With this configuration, the position is read in a mere 17 microseconds.

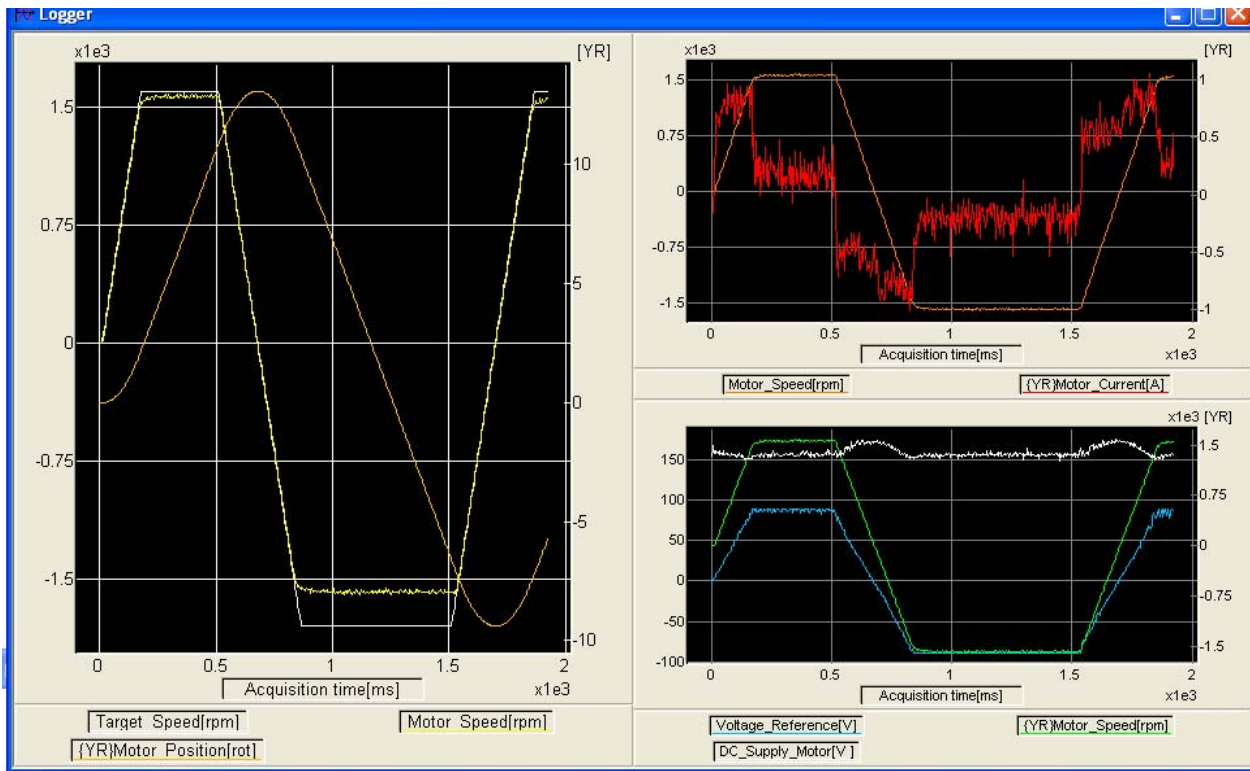
The Technosoft software allows extensive diagnostics of the SSI interface. Lower level variables can be viewed directly:



The “posSSIlng” variable contains the Gray code value in hexadecimal code (or any other notation). For 17-bit operation, this values ranges from 0x0000 to 0x1FFF. However, because of the Gray-code, it does not increment by 1 as the motor turns, The “posSSlbin” variable is the actual single

turn absolute position value, after translation from Gray code to binary. The "apos" variable is the actual position counter, which tracks the overall, multi-turn position.

With the SSI interface and position read out fully functioning, the motor can be commutated and velocity and position loops can be enabled and tuned. The following scope picture shows some of the internal drive variables during a move:



Although this specific application and implementation used a single turn absolute encoder, everything applies to a multi-turn encoder as well. Also, the resolution can be further increased to 18-bit or 262,144 counts per revolution.

In conclusion, the all-digital absolute encoder interface provides a high resolution, robust, and easy to implement solution for servo motor feedback. The use of metal instead of glass discs further increases the robustness relative to temperature, shock and vibration.

For more information about any of the above topics or general questions or comments, please contact us:



Motion Designs
contact@motion-designs.com
Tel 805.504.6177

Motion Designs is a technical sales and engineering company with extensive machine and motion control experience. We work with some of the best manufacturers in the industry as witnessed by our present line card:

- www.arcus-technology.com: Arcus Technology manufactures stepper motor, drive and controller technology, providing USB, Ethernet and Mod-Bus connectivity.
- www.ctc-control.com: Control Technology is a leader in automation control technology with extensive network connectivity capabilities.
- www.ixxat.com: Ixxat is a leading supplier of products and services for the data communication in industrial and mobile systems.
- www.stegmann.com: Stegmann is a leader in high performance motor feedback solutions.
- www.technosoftmotion.com: TSM is a leading DSP motion control technology company specialized in the development, design and manufacture of digital motor drive products and custom motion systems



T E C H N O S O F T
M O T I O N T E C H N O L O G Y