Using an Operator Interface Terminal (OIT) With a S200 Position Node Drive

Introduction

Many machine manufacturers find it useful to use an Operator Interface Terminal (OIT) with a single axis motor/drive system. This white paper discusses some of the issues in developing an a customized OIT graphic interface with the S200 Position Node Drive as our team decided to choose an OIT and configure it for a wide range of machine configurability.

The task

Our team set out to design a system that allowed easy configuration of a precision linear positioning system. We decided that the S200 Position Node drive could be pre-configured to perform virtually all of the main tasks. We wanted to allow the user to be able to adjust certain performance characteristics such as stop positions, speeds, ramp rates, etc. An off-the-shelf Operator Interface Terminal seemed like the perfect solution.

OIT Requirements

The S200 Position Node uses Modbus serial communications as a slave device. The user can read or write 32 bit registers and the data can be formatted as long or float. Therefore, the OIT must have 32-bit read/write access capabilities, a graphical development environment, Modbus Master support, RS232 port (38,400 or 19200 Baud), and support macros or scripts. We wanted simplicity, reliability, and the ability to communicate to the user as much information as possible to be unambiguous as to what to do. A CRT touch screen unit seemed to be the logical choice. Since the S200 Position Node uses a Mobus protocol and there are many OITs that support this protocol we felt confident to move forward.

Modbus Communication With an OIT

One of the first steps in developing an OIT is to communicate with the drive was to research capabilities. We found some of the terminology a bit confusing but easily made the connection. Terms such as PLC (device it is communicating with), Interval Packets (number of registers addressed per communication), and Station Number (Modbus address) were quickly understood. We needed an OIT with Modbus that could support the following requirements:

Communication Interface	: RS232
BAUD rate	: 38400 or 19200 BPS
Bits/character	: 8 bits
Parity	: Default Even (Settable to odd with S13-3 => Down)
Start bit	: 1
Communication Type	: Modbus RTU Master
Addressing	: This is also referred to as the Modbus address.

Set this to the corresponding address on the drive (PLC).
This is set on the drive through switches S11 and S12 for address 0-99.
(Station Number = S12*10 + S11)

Interval of block packs: 0

The “interval of block packs” is the number of commands that are combined and sent to the drive (PLC) in one message. The S200 position node drive will only process one command at a time. The value of zero tells the OIT not to combine any message together, but rather send each message out individually. Stated another way; The S200 Position Node supports only one register read or write per Modbus command.

Modbus ProtocolDiscussion

Using the OITs made Modbus communications easy and we found that we didn’t need to understand the protocol and become experts at it. However, a basic understanding of the Modbus protocol was found to be very helpful.

Modbus is a request/reply (sometimes called client/server or master/slave) protocol. The S200 Position Node acts as a server and will not communicate unless asked to. Therefore for every request the OIT will wait for a response. No other requests will be sent out until the pending response is received or a timeout has occurred. If a timeout occurs then the OIT will generate a communication error. What the OIT does with the communication error will depend on the OIT and how the developer configured it. A common action on a communication error is to display a popup window (see Figure 1). The PLC terminology refers to the S200 Drive.

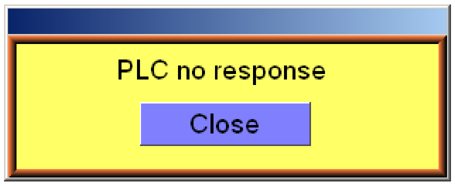

PLC no response

Figure 1 An Example Popup window On A Communication Error

Modbus is an application layer (OSI 7) messaging protocol.[i] Our implementation of Modbus can be defined simply by stating that a Modbus communication from the OIT can either read a register or write a register. To do this, Modbus has defined a protocol data unit (PDU) as shown in Figure 2. The PDU is made up of an address field (Addr), a function code (FC), the data (Data), and a checksum (CRC). The address refers to the identity of the device (the Modbus address of the S200) that you wish to communicate with and is set on the S200 drive through switches S11 and S12. The function codes will be discussed in the “Function Codes” section. The data is the information sent/received from the drive and the CRC is the calculated checksum. The checksum will be calculated for the developer by the OIT design software and we need not concern ourselves with the method used to calculate the checksum. However, interested reader can refer to the sited reference[ii] for details on the CRC checksum.

Addr	FC	Data	CRC

Figure 2 Modbus Serial Line Protocol Data Unit (PDU)

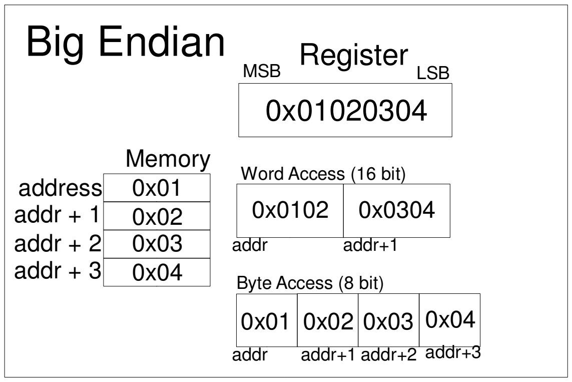
The Modus data field consists of 3 components; The register address of the variable inside the S200 PN that you wish to access, the number of data bytes that are being worked with, and in the case of a write request, the actual data being written to the register. Thee S200 Position Nude has bot variable names and a Modbus register association. This is well documented in the S200 Position Node User's guide.

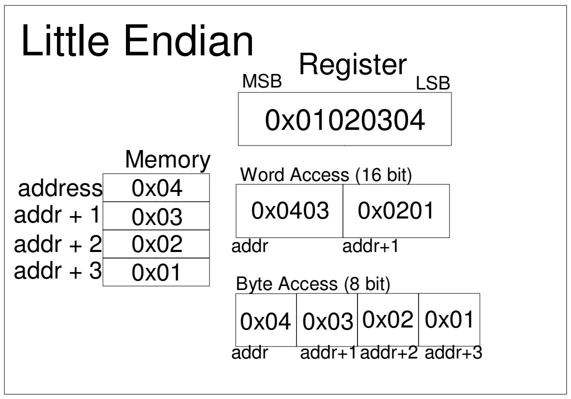
Function Codes

The Modbus protocol uses function codes (FC) as commands.[i] To communicate to the S200 drive only the public function codes that we will use are the "Read from Holding Register" (FC=3) and "Write to Holding Register" (FC=16).

Register Addressing

There are two common register (or memory) configurations. The first memory configuration places the most significant byte (MSB) in the lowest address space (big endian – see inset), while the second memory configuration places the least significant byte (LSB) in the lowest address space (little endian – see inset). The reason this matters when developing an OIT is depending on the drive and the OIT memory architecture we may need to swap words in order to get the correct data. The OIT/Drive that was employed in this white paper required a word (16-bits) swap. In the OIT that we selected we found that we could accomplish this by using a 5x command instead of the 4x command. The 5x command swaps the words for us, while the 4x command does not. You may need to perform the word swapping in a macro or script if the OIT does not support this function.





The S200 drive contains 32-bit registers and Modbus uses big endian, 16-bit words access. So in order to access the S200 drive a double 16-bit access with a word swap is required. Using the word swapping command

(5x command) the Modbus address needs to be increased by one. The OIT decrements the Modbus address by one and then retrieves the two 16-bit words performing the word swap. An example is listed in the ***OIT Jog Example*** section.

Parameters and Commands

The S200 Position Node has all features mapped to registers. Some variables can be used to adjust performance and are called variables or parameters, such as speed. Other registers cause the drive to execute a function, such as STOP, and are called commands. An abbreviated S200 position node command table[iii] is listed in Table 1. These commands are used in the ***OIT Jog Example***. The full command table is presented in the S200 Position Node User's Guide[iii].

Table 1 S200 Position Node Drive - Abbreviated Command

Command Name	Modbus address	Description
STOP	508	1 = Force velocity to 0
ENABLE	2404	1=enable drive 0=disable drive
MJOG	290	Jog at VJOG speed and ACCR/DECR Ramps
VJOG	578	Jog command (velocity) while in Motion Tasking or Gearing Modes

OIT Jog Example

This is a simple example (Figure 3) that allows the new OIT developer a chance to learn some basics of OIT / S200 Position Node Drive communication. This example enable/disable the drive, allows the user to set the jog speed, start jogging the motor, and stop jogging the motor. The "Disable" button sends a Modbus command (5x) to address 2405 (Modbus address + 1). Recall we need to add one to the Modbus address when using the word swapping command (5x), so the Modbus address from Table 1 (2404) turns into 2405. To software disable the drive the data that is sent to Modbus address 2404 must be a zero. The ***Enable*** button sends the same command to the drive as the ***Disable*** button except that the data is set to a one. The activation of the ***Jog*** button sends a one to address 579 (Modbus address+1). This jogs the motor assuming that the drive has been enabled. Set the jog velocity by writing to Modbus address 578 (remember to add 1 to the address when using the word swap command).

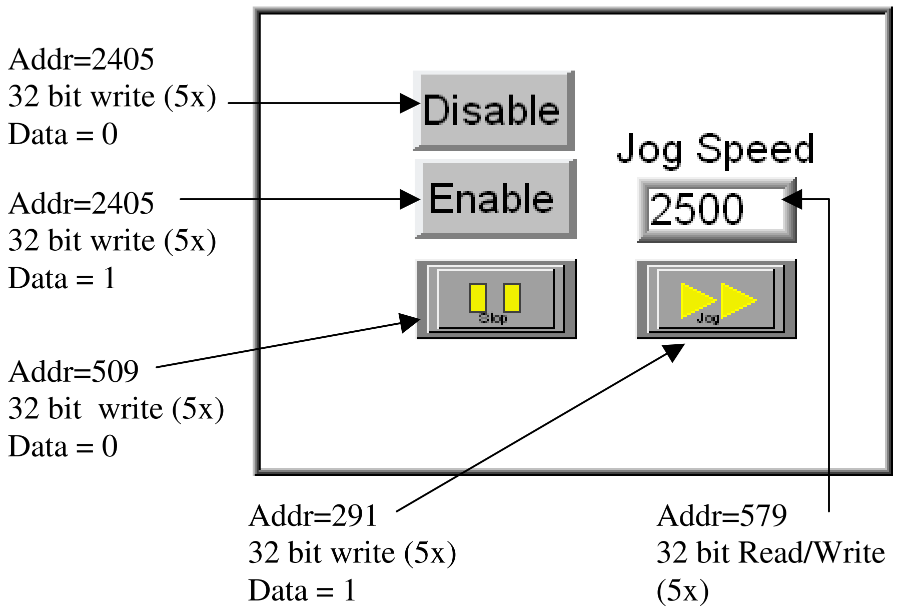


Figure 3 Operator Interface Terminal (OIT) Jog Example

Conclusion

Some machines can benefit greatly from having an Operator Interface terminal to allow the user to adjust some parameters of the machine performance. We set some goals of being able to modify such parameters. We selected an OIT Terminal and programmed it to perform these tasks. Foregoing the issues required to make the interface ergonomic for the user we discovered how the 'off the shelf' operator interface terminals operate and some of the advantages to certain models. Since the S200 Position Node is a configurable drive and not a programmable drive we found it advantageous to have some higher-level of features in the OIT. Modbus communications and register sets made the task simpler but we did find a few challenges such as numerical processing capabilities of the OIT. Our mating an OIT with the S200 Position Node drive was successful in every goal set.

References

http://www.maplesystems.com/products/silverseries/silverseries_ezw.htm

[i] MODBUS Application Protocol Specification V1.1b, http://www.Modbus-IDA.org, December 28, 2006.
[ii] MODBUS Over Serial Line Specification & Implementation Guide V1.0, http://www.modbus.org, December 02, 2002.
[iii] S200 Position Node User's Guide Revision B, http://www.danahermotion.com, Oct. 23, 2007.