Capturing Live Data

Malcolm Moore

Live Trade Signals

One of the outputs from stock exchanges is an electronic message of the trades as (or very soon after) they happen. This electronic message has a worldwide standard format called the 'e-signal' (short for electronic signal). This document does not describe the structure and format of the e-signal apart from the notion that this signal is in effect a datagram packet that looks like a database table record that contains many fields, including the symbol, time, price and volume details (of which are primary importance) in this document.

The e-signal also contains several other details about the source and sink of the trading seats involved and other details which may be useful for deeper forensic analysis, but not for direct tracking of trade prices and volumes as required for normal live trading activities.

There are two areas that need to be considered, the distribution and capturing of live trade signals, and the process to make these live trade signals into practical data that can then be analysed into useful information.

Before the live trade data is captured, it is useful to understand some of the possible data broadcasting / transmitting datacasting transmission techniques that could be deployed to distribute this data. Some datacasting methods are naturally very inefficient, and with a little bit of lateral thinking these datacasting methods can be made extremely efficient to the point that in some cases there are massive savings in network infrastructures and traffic densities.

Experience has shown that simply using the tick data as the datum to work with is seriously flawed, because on a more strict arrangement tick data is not time bound, and it is essential to have trade data that is time bound by 'timeslots'. Unfortunately because of the wide range of uses for trade data, (which is field intensive) the transmitting of 'candlestick' data (symbol, date, time, open, high, low, close, volume) in pre-determined time slots would void most of the ancillary fields used for further analysis, so it is necessary to transmit the entire trade data. What has to be understood is that in live terms a typical trade is not a 1:1 match of the volume of securities and typically there are three 'transactions' (or trade messages) to make up one physical trade, so the volume of trade messages (or signals) can be quite intense.

How Trade Data is Distributed via Analogue TV Transmissions

In Australia, (until August 2007) the e-signal was transmitted through the SBS TV Channel in cooperation with a company called MarketCast using a method was particularly efficient as it involved only one transmitter per unit area, covering millions of premises in every capital city and several regional areas.

In practice the live data was streamed as consecutive datagram packets in the analogue TV channel via the Vertical (Synchronisation) Insertion Test Signal (VITS) time slots, which allowed about 20 horizontal lines of data per frame where the picture is nominally black, while the vertical analogue sweep 'retraces' to the bottom of the picture (raster). As there are 50 frames per second in the Australian Phase

Alternating Line (PAL) system, there are about 100 lines of data that can be synchronised into the horizontal scans.

In the Australian analogue format, the line frequency is 15,625 Hz, which means that each line takes 64 usec. As there is about 12 usec in every horizontal 'retrace', there is about 52 usec left over for the visual horizontal trace. Assuming there is about 50 usec available per horizontal line and about 20 VITS lines per frame, and 50 frames per second; this means there is about 50 milliseconds of synchronised time per second.

While this time allowance might appear short it also has to be understood that the usable bandwidth is at least 1.5 MHz and the signal to noise ratio is typically greater then 24 dB, meaning that a 4 bit per say 800 kHz system should run with very few errors. If the data is transmitted as a bipolar stream, the picture would remain nominally black and the vertical synchronisation is not unduly affected.

Say for example the base (carrier) frequency is 800 kHz, and the signal to noise ratio is say 24 dB then there is a nominal 4 times 6 dB or 4 bits per Hz, and this works out at about 3.2 Mb/s. Assuming there is some spare this would work at say 2 bits per Hz = 1.6 Mb/s. Allowing 50 msec per sec * 1.6 Mb/s = 80 kbits/sec and although this is not 'Broadband' rates, the data is significantly fast enough to transmit live data with ease, even if say 50% were lost in overheads, the useful data rate is still 40 kbits/sec.

Assume that each trade message (datagram packet) takes 200 bytes of data, (or 1600 bits at eight bits per byte), then this system is capable of transferring about 25 datagrams per second, or about 250 datagrams in 10 seconds, 3000 datagrams per minute, 15000 datagrams per five minutes.

This data transfer capability should be substantially more than what is needed, but there is concern that at the opening of the market, the data transfer capability may not be enough because the possible very high volume of trades at the open.

How Live Data could be Distributed over the Internet

If the e-signal were transmitted via the Internet to several thousand customer terminations, then a very high proportion of the backhaul Internet infrastructure would be 'streaming' the same data through thousands of backhaul channels and this will significantly congest the Internet backhaul infrastructure. The live data streaming would also last for at least six hours per day and the per customer data rate would be about say 50 kbits/sec or about 22 Gbits/month, and that makes for very expensive Internet usage.

One internal solution could be that a smart telecomms engineering could deliberately distribute the e-signal to every district exchange site at a virtual port in the district router. This rather smart engineering approach would be very efficient in network terms (but I doubt that any telecomms engineering have considered this network topology.)

The signal would occupy one part of a network as a singular stream, instead of hundreds of singular streams, and the savings are very obvious. At the local exchange, the terminal Optical Line Terminator (OLT) could directly 'virtually switch' into this port and provide this data with very little network overhead.

Looking at this thin streaming approach in another way, the signal could be directed back to a nearby port that triggers an alarm if the signal ceases. This back-checking

approach would provide another level of network integuity to prove the operational performance of the network using live data, and prove the live data through to the terminal exchange on a 24/7 basis. With this method, there is a very little demand on the network and it uses data that is already sold on a wholesale basis to potentially thousands of customers.

Data Load Distribution

Experience has also shown that end-of-day (EOD) works very well because it is commonly time bound on day episodes, but the time-bound data comes in at the best 24 hours too late – but in most cases this is not a problem as prices peak and trough over some days. The end result is that by using EOD data the price has to rise more than say 5% before the rise is noticed, and similarly thew price has to fall by at least 5% before the fall is noticed.

The real problem is that the fall rate (gradient) is at least twice the gradient of the typical associated rise gradient, so the chance of identifying a fall and acting on it is somewhat remote without say a 10% drop happening. Considering that the piece 'step' might be typically 60% then this is dropped to a typical 45% gain, or about 30% of the 'step' is lost through using EOD data.

When Live Data is used, the EOD data has to be considered in terms of 'live data', where the EOD period is typically 6 hours (10 am until 4 pm). If the time periods are reduced to 30- minute time slots, then much of the early trade is lumped into the first trade of the day, and a huge amount of trades are recorded in the first 30 minutes. By moving towards real time, if a 5-minute time slot is used, then the high volume of early trades as the market opens are then shared between at least six time slots (if not more), and the trading density is considerably smoothed.

Consider also that the stock market does not open all at once as it is alphabetically staged in, through several five-minute intervals. With this five-minute approach, the size of the stock volumes is not too large and the trades are time bound. A well-hidden advantage is that the five-minute candlesticks are rather tight and in this, there is little variation in the candlesticks, so this means that there is little approximation required to get an average price and the known volume over these five-minute periods.

A slight variation is that the EOD data is used to 'get-close' to the results using EOD data and then 'move in' using five-minute time slots to hit on the time within say 20 minutes of the critical price, and that means the buy-in price will be in the order of 2% off the low floor and within 3% of the peak price, meaning that instead of losing about 30% of the total 'ramp' the loss is typically 5%, and considerable whip-lashing is eliminated.

To get the five-minute time-bound candlestick data, the e-signal has to be read and this data buffered and saved into timeslots for analysis, and here is one way to do it:

Read E-Signal Process

Assume this e-signal comes in via the USB port, very high priority. This is a special area and a "USB Driver" needs to be written to interface with the USB modem. Alternatively, the 'modem' could be interfaced with Internet (probably as Cat-5) so that it can connect as a LAN component, and in either case this interface will also need a driver so that it seamlessly interfaces with all types of PCs Macs etc.

On e-signal being received, de-scramble e-signal and put the data into the live buffer. The descrambling process is to identify that the signal sent is a valid e-signal, and then relate the datagram contents onto the necessary fields, call the buffer and load the fields into the buffer, then reset the routine and wait for the next e-signal.

How to Program in a Timer Sequencing Process

This Timer Sequencing process sets the active time window so that the trigger is conveyed to cause the reading of data off the buffer into the 5-minute array only during this window. There are three sequential gates that need to be satisfied before the Use the Date function to identify active Weekdays – Monday through to Friday as these days are the only days where the process continues.

Use the Time function to identify active window in time, (after 9.45 am and before 4.35 pm Mon-Fri)

At the 5-minute time, this timer is to trigger the sequence of: Zeroing the 5-minute array, Zeroing the immediate array, Reading buffer process.

The Process of Reading the Live Data Buffer

This process to read the buffer carries a lot of background work to get the buffered live data into time-bound data slots and warehouse saved into MetaStock form so that the analysis process can take place in a virtually seamless data environment.

The live data comes in from a stock exchange in a random timed order, based on the completion time of trades (and most of the time during trading hours). This live data can be likened to single lane road traffic where vehicles are positioned in a rather random timed function. The application peripheral interface (API) associated with receiving the live data will separate out datagrams that are not trade data, and pass trade data through with a ‘flag’ to say that Live data is ready to be warehoused.

From experience a cyclic First-In-First-Out (FIFO) buffer is absolutely essential and this buffer needs to hold at least about 2000 records, so the buffer can hold the onslaught of trades that happens as the market opens. The Live data usually comes into the API from the modem or LAN in the form of comma separated variable (CSV) strings or specific API variables. It is highly convenient to either leave the string as it is and simply load it onto the FIFO buffer, or transfer the contents of the APIs’ variables directly into the FIFO buffer array so these values can later be quickly transferred from the FIFO buffer as required.

A software buffer can easily read more than 3000 records per second, but the real art in engineering a FIFO buffer is to arrange the record pointers so that they never collide, and that the records never get corrupted especially if the reading pointer is adjacent to the writing pointer.

The FIFO buffer also needs internal flags to indicate if data is in the buffer. In reading the next row of records from the cyclic FIFO buffer into the immediate array it is wise to check and find out if the symbol in this record is identical to the symbol used for the last record.

The reasoning is that most trades take more than one live data record, and typically take three records to get the total number of securities to effect a trade. With this logic, if the symbol is new for the last record, then there is about an 83% chance that the next record will be that same symbol and about a 66% chance that the following record will have that same symbol, and this coding arrangement dramatically reduces

processing time. The Visual Basic code that is written below shows how the FIFO buffer can be constructed and how the records can be taken from an API (in this case the MarketCast API), loaded into the FIFO buffer and then translated into an immediate array for transfer into a 5-minute timeslot.

```
'Run the Data Buffer
On Error GoTo BufferErrorHandler
If lngInCount > 2000 Then 'Test the Input Counter placement
lngStatus = 1 'The InCount is now under the OutCount
lngInCount = 0 'Reset the Input Counter
End If
'BufferReady:
lngDataReady = 0 'About to put in new data
lngInCount = lngInCount + 1 'Increment the Input Counter
strBufSymbol(lngInCount) = strTradeSymbol
sglBufPrice(lngInCount) = sglTradePrice
lngBufVolume(lngInCount) = lngTradeVolume
strBufEpochTime(lngInCount) = strTradeTime
strBufMMDateTime(lngInCount) = strMMDateTime
sglBufAskPrice(lngInCount) = sglAskPrice
lngDataReady = 1 'Data loaded and buffer is now ready
lngDataAvailable = 1 'The Buffer is carrying data to be read
out
'Get the next recordset Out from the buffer
If lngDataAvailable = 1 Then 'New Data is available
lngOutCount = lngOutCount + 1 'step the Out Counter on to index new
buffered data
If lngOutCount > 2000 Then
lngStatus = 0 'Normal Status 0 : Starting over!
lngOutCount = 1 'Start from 1
End If
If lngStatus = 0 Then
If OutCount < InCount Then
ReadData 'Read the Data from the buffer
ElseIf OutCount = InCount Then
lngDataAvailable = 0 'No more data is available after this
one
If lngDataReady = 1 Then
ReadData 'Read the Data from the buffer
End If
End If
ElseIf lngStatus = 1 Then 'Status = 1
If lngOutCount > lngInCount Then
ReadData 'Read Out the data from the buffer
ElseIf lngOutCount = lngInCount Then
lngDataAvailable = 0 'No more data is available after this
one
If lngDataReady = 1 Then
ReadData 'Read the Data from the buffer
End If
End If
End If
'NoMoreData 'hop over the read data area
'ReadData:
strBufSymbol = strSymbol(lngOutCount)
sglBufPrice = sglTradePrice(lngOutCount)
lngBufVolume = lngTradeVolume(lngOutCount)
strBufEpochTime = strTradeTime(lngOutCount)
strBufMMDateTime = strMMDateTime(lngOutCount)
sglBufAskPrice = sglAskPrice(lngOutCount)
'NoMoreData: 'The last data has been loaded
'The next available recordset has been retrieved from the buffer
```

Although this code is poorly written and poorly structured, it does show how the FIFO buffer works and how the MarketCast API (or any other API) can be interfaced.

Note that SQL programming can make this next process to transfer the immediate array into the 5-minute timeslot records much faster as there are already machine code routines that are optimised to do these data finding, appending and saving functions very quickly.

Load the MetaStock Files

On the 5-minute clock: Copy the 5-minute array into a secondary 5-minute array, then clear the 5-minute array for loading from the buffer. (It might be faster to directly load MetaStock 5-minute data than transfer and load the MetaStock data).

Use the Metadata API to load the MetaStock data from the secondary 5-minute array, then clear the secondary 5-minute array.

Allow the buffer to start loading buffered e-signal data into the 5-minute array again.

Note that SQL programming can make this process much faster as there are already machine code routines that are optimised to do these functions very quickly.

Why Trade Data needs to be Time-Slotted

Trade data provides the facility for intra-day decisions, while EOD data provides the facility for much longer-term trading decisions, and it therefore follows that EOD data has the facility to determine when to change a position between buy / hold / sell, but that decision is at least a day late. If 'Tick' data (individual trades based on a time continuum where the exchange is open for trading) is used, then moving averages can be step-shifted with the Tick data trades (irrespective of volume) but this inevitably leads to over-reactive moving averages, which in turn leads to false calling on trading positions.

EOD data has several built-in safety catches that inherently normalise the EOD (6-hour) time slots. The EOD time slots includes the rush of trades at open and at close, and through this the EOD data inherently levels out the volume to a large degree so much that many indicators operate quite well without taking into account the volumes. Trade data in shorter than EOD timeslots does not have these safety catches built in, so it is necessary to build and implement these safety catches so that trade data with much shorter timeslots is compensated by volume weightings so that the moving averages can be proportionally weighted to reflect the relative trading volumes.

Almost all active securities trade every trading day, and with this assumption, there is a new trading price at the close of every EOD timeslot to be analysed. Again, most technical analysts are aloof to this problem and ineptly assume that there is a trade every day. Most data analysis packages count back a certain number of trading days and do their SMA maths, or in the case of the EMA calculation, if there has been no trade that day, then the value is null and no calculation is done (or the previous close price is repeated).

From my perspective with an SMA if the trade values are null, then the SMA count has to continue back until the count is complete. If it is a 50SMA then the SMA routine needs to count back through 50 traded days (timeslots), even if that means counting back through say 126 days (timeslots) to get the 50 count!

If it is an EMA then the maths is considerably simpler, as the EMA will only calculate if the value is not a null! As MetaStock data does not hold 'null' records, the routines for calculating SMAs and EMAs are rather straightforward.

With the prices, volumes etc in an class array (where each dot sub-class has its own data formatting), it is very easy to load an array from a select MetaStock symbol, make routines that step through the array and calculate averages, weightings and more complex indicators. What is not so obvious is that with timeslots that are less than EOD periods, it is necessary to include weighting to balance the effect of trade volumes with prices.

How to Weight Moving Averages

To get a relative weighted volume figure it is first necessary to create a 10SMA of the volume (five days in the week to nullify the weekly ripple), and use this 10SMA as the average volume weight. With EOD calculations, the last day's volume is an indication of the weight of the moving average, so:

Volume Weighting = Last Day Volume / 10SMA Volume

A Weighted EMA and Weighted SMA would have the leading formulas:

EMA(Weighted) = EMA /Volume Weighting

SMA(Weighted) = SMA /Volume Weighting

For example say the 10SMA Volume is 10,000,000 and the last days Volume is say

12,000,000, then the Volume weighting is 12/10 = 1.2

A 55EMA (unweighted) would change to 55 / 1.2 = 45.833EMA for this day's calculation. While this might seem harsh, remember that the weighting is normalised against the 10SMA of the volume, so the longer-term moving average would not be altered from its nominal value. In practice this weighting for EOD indicators would even out fairly quickly, but if this were moved into intra-day timeslots, then the weighting can be much heavier – especially at the open and close of the trading day.

Say the time-slots are placed on the hour, so the unweighted average would be 1/6 the moving average day trades, so a 30SMA would become 30 * 6 = 180SMA for each of the 6 time-slots, and each of the time-slots would have a weighting of 1/6 (0.16666) of the total volume. If, say 30% of the total day's trades happens in the first time-slot, then the weighting would be 0.3 * 6 = 1.8, so the 180SMA would change into a 180 / 1.8 = 100SMA to reflect the weight of the trades in this time-slot.

While the maths for all this weighting of moving averages might appear rather messy, it does however take into account the relative volume of the trades during the timeslot, so if there is a run on the market, the moving averages will quickly shorten their time constants and cause different moving averages to step over one another earlier than later, and conversely; if the market is trading lightly, then the time constants will be stretched out so as not to cause a false trigger from over-zealous moving averages. The good news is that once the maths is programmed in then these activities work seamlessly in the background to optimise the best trading opportunities!

This situation makes the EMA an indicator of choice because if the volume is greater then zero, then the EMA will be

How to Normalise Moving Averages with Smaller Time-Slots

One line of thought is to use EOD data-based indicators and when the decision is becoming imminent, move over to Trade data-based indicators to make the position decision with a minimum of delay. Part of the MetaStock Data includes Open, High, Low and Close prices, but in reality these extra data values are not necessary, as the only real data is the Average (or 'spot') Price over the past 5 minutes and the Volume. Putting these two together, we have 'spot' prices, along with the volume, so we could use volume weighted 'spot' prices. The alternate beauty is that the candlesticks are so short that there is very little approximation, and the spot prices will be very close to the real prices.

When using trade data, put into say 1-hour timeslots, then the close price is the last price of that hour and then analysis takes place; so in this case there will be eight timeslots per day, and seven places where trading action can be taken during that day. What has to be realised is that if the EOD analysis is using say a 20SMA; then the direct equivalent 1-hour analysis would have to use 20 * 8 SMA = 160SMA, so that the curves would follow the same (or extremely similar) paths. If 5-minute timeslots were being used, then there would be nominally 96 timeslots per day so an equivalent 20SMA would blow out to a 1920SMA, making the maths somewhat clumsy and comparatively slow.

It should start to become obvious that calculating a 1920 SMA or much larger SMA for example if the EOD 107SMA was converted into the 5-minute SMA arrangement then the 5- minute timeslot equivalent would be a 10272SMA, and this would quickly become highly impracticable, so there must be better ways.

Most technical analysts would immediately jump onto the EMA to replace the SMA because calculating the EMA is extremely easy. To calculate the SMA it is necessary to have a data array that holds (in the trade data case) tens of thousands of data records so that results can be calculated.

To calculate the (first order) EMA all that is needed is the last EMA result, the EMA period, and the new price! It is no wonder that most technical analysts ineptly choose to base most of their analysis on the (first order) EMA indicator! Unfortunately, the transient response is the critical factor that literally determines the decision for the trading position, and the EMA consistently gives very inconsistent (and slow) results because of the long exponential tail.

In the earlier documents about low-pass filtering, it became fairly clear that the EMA has a far less than satisfactory transient response from a step excitation, the SMA can be timed as a multiple of the weekly oscillations to self-cancel weekly ripples (where the EMA cannot), but the SMA is rather difficult to manufacture if tens of thousands of 5-minute timeslot figures need to be considered. Very fortunately, a second order low-pass filter can be constructed, and it only has one previous transient state that needs to be calculated, and the transient is significantly better than the first order EMA and nearly as good as the SMA, but without the massive array calculations.

How to Link Trade Data with EOD Data

A very good hybrid compromise is to use the EOD to bring the SMA calculations up until about 20 (or a few less) days behind today, and from that point the existing low-pass figures can be used with 5-minute timeslot data. This way the SMA (up to a few weeks ago) can be calculated very quickly based on EOD data, and seamlessly provide the transition into the much smaller time-slot data without resorting to massive SMA calculations everywhere.

The hidden benefit of this hybrid approach is that the size of the Trade Data files can be kept relatively small as they only need to be kept for say three weeks, and all the older 5-minute data can be cleared out, leaving a MetaStock-based Trade Data database that would typically be less than say 50 Mbytes.

When it comes to candlestick graphing the recent data – if the displayed time span is greater then say three weeks, then it makes sense to display the EOD data, but under three weeks, as the Trade data is readily available then the graph can revert to (say) five minute timeslots, and the detail in the daily trading will become obvious.

Candlesticks are excellent for seeing the spread of trade prices over a given time span but it has to be realised that with the heavily summarised data that is held in the candlestick that absolutely no assumptions can be made into the trajectories of the trades during these time spans. With this understanding, the shorter the candlestick and its wicks, the less the approximation, and this is why Trade data can create candlesticks over much shorter time intervals then EOD data. As these time-slots (or intervals) are reduced, there will be more time intervals that have no trading.

These blank intervals will visually distort the trends and appear to visually confuse some indicators, and this is another reason why a compromise is necessary so that the time-slot (intervals) are made long enough to capture trading in most time-slots, and the intervals made short enough that the candlesticks are themselves short so that pricing approximations are minimised.

As far as establishing an analysing program is concerned, the first stage could analyse up until say three weeks ago using EOD data and save those results, and then switch over to Trade data-based timeslots for analysis within the last say three weeks. From this point only the 5- minute timeslot data needs to be saved as larger timeslots can be quickly ‘manufactured’ as multiples of the 5-minute time-slotted Trade data.

The difference with using Trade data and shorter intervals than EOD data can allow is that the moving averages will step over each other most likely during the trading day and not the day later, so that trading decisions will happen more often than not during the trading time. The above example would have eight time slots during the day, and the volume weighting of the grouped trades could make all the difference between selling and delayed holding positions.

How to Make Trade Indicators Predictable

It has always been said that the best prediction of the future is to look at the recent history, and this is the reason why so many trading opportunities have been missed – and the less than recent trading history shows classical examples of missed opportunities.

Technical trade data is amazingly transparent to market activities – because it actually shows what is going on (with relatively inside trading) before the financial reporting happens. Sure, the securities will not be traded to or from people in the businesses (like directors), but relatives will be there trading on good inside advice, and technically, these trades are seen but usually not enough to make an indication with.

The future cannot always be accurately predicted, but with technical historical data, it is quite plausible to extend the current trajectories by about five time-slots if you have no less than about 50 recent time-slot data records to work with (and the size of the candlesticks is both consistent and small).

Straight line approximations of the various prices and indicators curves is crude but functional, and if the order of the polynomials to approximate the curves is raised to about 4 or 5, then the polynomials can really hug the existing prices and be extrapolated a few periods to get a reasonable result that will predict with a good degree of accuracy of an impending change of state between buy / hold / sell.

The values for Open, High, Low, Close, Volume, SMA, EMA, MACD, Trigger etc are in reality time clocked step values, while the polynomial that approximates these stepped values is really a continuous curve that approximates the stepped values at the clocking edges. It is very important to comprehend this structural difference and realise that the polynomial should not be used as an inter-time differentiator for changes in trading positions without reference to the clocking edges.

Assume this predictability is about 90% accurate, then extrapolations with EOD data and associated information as EOD indicators will give about 4 to 5 days forewarning that a change of state is imminent and this gives ample opportunity to act on the position and maximise the profitability by placing a sell or buy order that will very likely trade very close to the crest or trough with a reasonable degree of safety before the price velocity changes significantly.