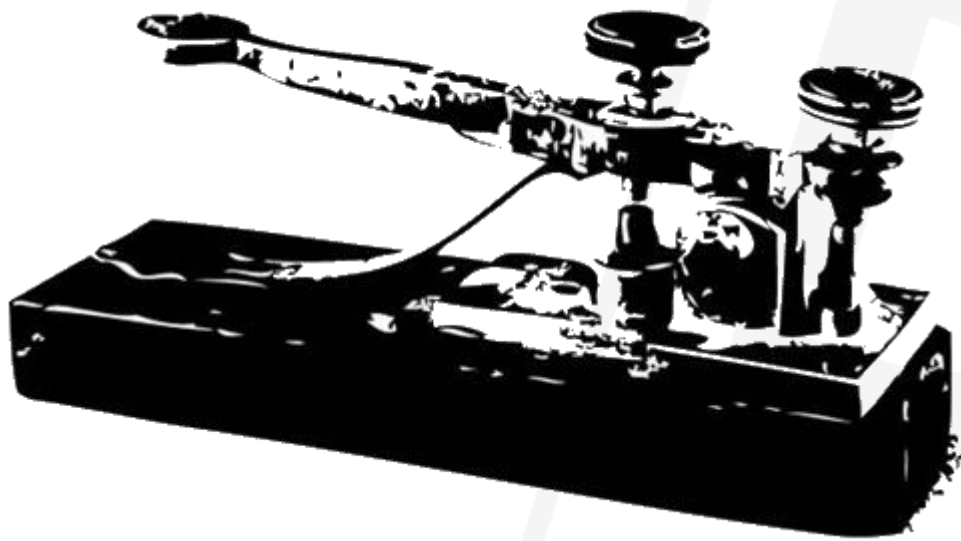


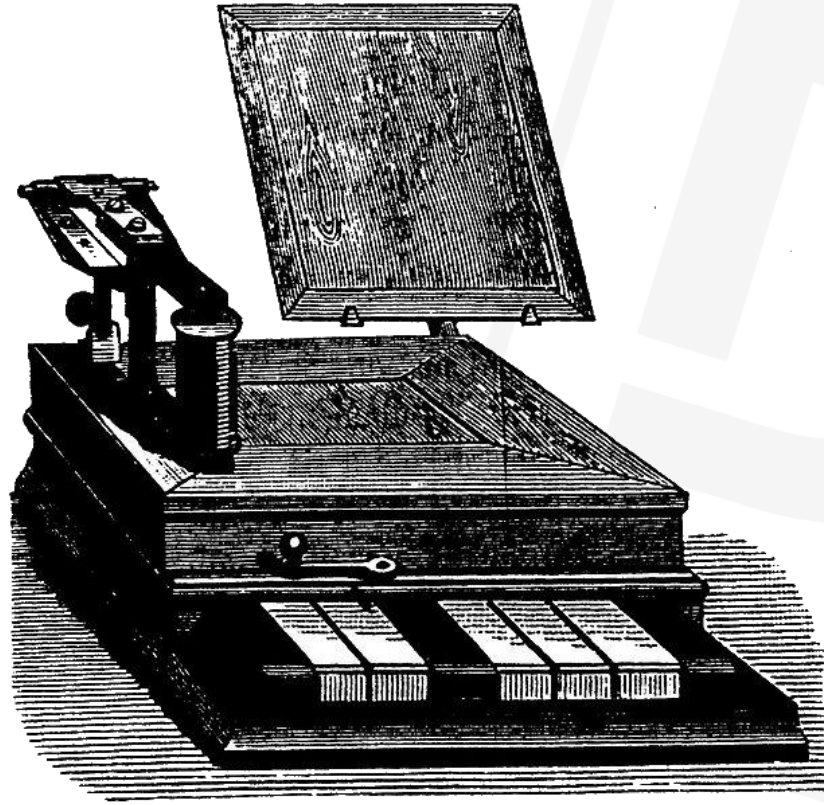
DataFlex NextGen

Presenter: Harm Wibier

SYNERGY 2019
CRUISING TO NEW HORIZONS



W H A T H A T H
G O D W R O U G H T



SYNERGY 2019 CRUISING TO NEW HORIZONS

J. M. E. BAUDOT.

PRINTING TELEGRAPH.

No. 388,244.

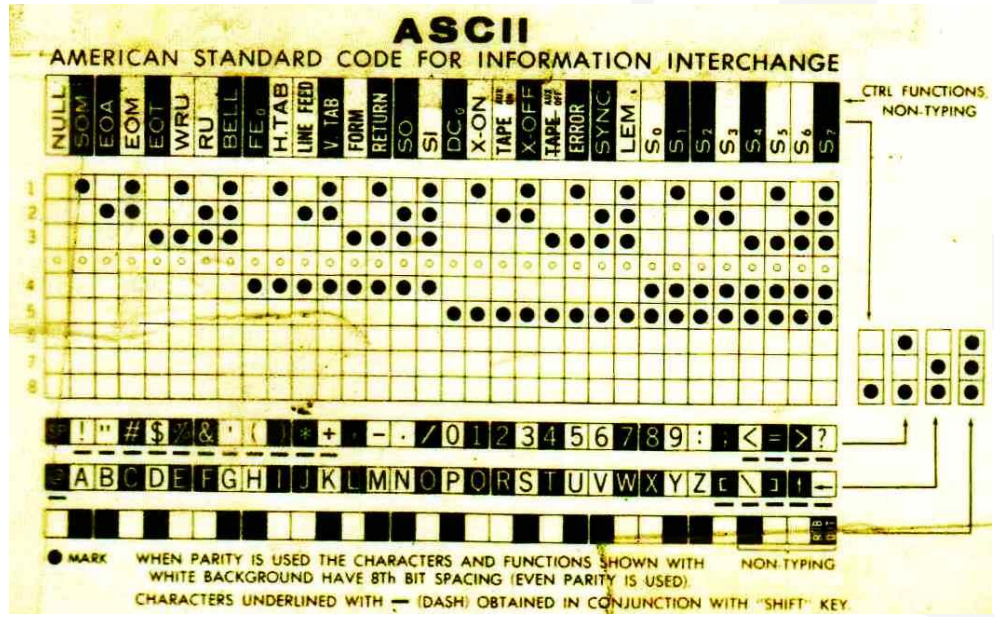
Patented Aug. 21, 1888.

Fig. 24.

	1	2	3	4	5
A	+	-	-	-	-
B	-	-	+	+	-
C	+	-	+	+	-
D	+	+	+	+	-
E	-	+	+	-	-
F	+	+	+	-	-
G	-	+	+	+	-
H	+	+	-	+	-
I	-	+	+	-	-
J	+	-	-	+	-
K	+	-	-	-	+
L	+	+	-	+	+
M	-	+	+	+	+
N	-	+	+	+	+
O	+	+	+	-	-
P	+	+	+	+	+
Q	+	-	+	+	+
R	-	-	+	+	+
S	-	-	+	-	+
T	+	-	+	-	+
U	+	-	+	-	-
V	+	+	+	-	+
W	-	+	+	-	+
X	-	+	-	-	+
Y	-	-	+	-	-
Z	+	+	-	-	+
·	+	-	-	-	+
·	-	-	-	+	+
·	-	-	-	-	+
·	-	-	-	-	-

INVENTOR:

Jean Maurice Emile Baudot



Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000	0000	00 [NUL]	32	0010	0000	20 space	64	0100	0000	40 @	96	0110	0000	60 `
1	0000	0001	01 [SOH]	33	0010	0001	21 !	65	0100	0001	41 A	97	0110	0001	61 a
2	0000	0010	02 [STX]	34	0010	0010	22 "	66	0100	0010	42 B	98	0110	0010	62 b
3	0000	0011	03 [ETX]	35	0010	0011	23 #	67	0100	0011	43 C	99	0110	0011	63 c
4	0000	0100	04 [EOT]	36	0010	0100	24 \$	68	0100	0100	44 D	100	0110	0100	64 d
5	0000	0101	05 [ENQ]	37	0010	0101	25 %	69	0100	0101	45 E	101	0110	0101	65 e
6	0000	0110	06 [ACK]	38	0010	0110	26 &	70	0100	0110	46 F	102	0110	0110	66 f
7	0000	0111	07 [BEL]	39	0010	0111	27 '	71	0100	0111	47 G	103	0110	0111	67 g
8	0000	1000	08 [BS]	40	0010	1000	28 (72	0100	1000	48 H	104	0110	1000	68 h
9	0000	1001	09 [TAB]	41	0010	1001	29)	73	0100	1001	49 I	105	0110	1001	69 i
10	0000	1010	0A [LF]	42	0010	1010	2A *	74	0100	1010	4A J	106	0110	1010	6A j
11	0000	1011	0B [VT]	43	0010	1011	2B +	75	0100	1011	4B K	107	0110	1011	6B k
12	0000	1100	0C [FF]	44	0010	1100	2C ,	76	0100	1100	4C L	108	0110	1100	6C l
13	0000	1101	0D [CR]	45	0010	1101	2D -	77	0100	1101	4D M	109	0110	1101	6D m
14	0000	1110	0E [SO]	46	0010	1110	2E .	78	0100	1110	4E N	110	0110	1110	6E n
15	0000	1111	0F [SI]	47	0010	1111	2F /	79	0100	1111	4F O	111	0110	1111	6F o
16	0001	0000	10 [DLE]	48	0011	0000	30 0	80	0101	0000	50 P	112	0111	0000	70 p
17	0001	0001	11 [DC1]	49	0011	0001	31 1	81	0101	0001	51 Q	113	0111	0001	71 q
18	0001	0010	12 [DC2]	50	0011	0010	32 2	82	0101	0010	52 R	114	0111	0010	72 r
19	0001	0011	13 [DC3]	51	0011	0011	33 3	83	0101	0011	53 S	115	0111	0011	73 s
20	0001	0100	14 [DC4]	52	0011	0100	34 4	84	0101	0100	54 T	116	0111	0100	74 t
21	0001	0101	15 [NAK]	53	0011	0101	35 5	85	0101	0101	55 U	117	0111	0101	75 u
22	0001	0110	16 [SYN]	54	0011	0110	36 6	86	0101	0110	56 V	118	0111	0110	76 v
23	0001	0111	17 [ETB]	55	0011	0111	37 7	87	0101	0111	57 W	119	0111	0111	77 w
24	0001	1000	18 [CAN]	56	0011	1000	38 8	88	0101	1000	58 X	120	0111	1000	78 x
25	0001	1001	19 [EM]	57	0011	1001	39 9	89	0101	1001	59 Y	121	0111	1001	79 y
26	0001	1010	1A [SUB]	58	0011	1010	3A :	90	0101	1010	5A Z	122	0111	1010	7A z
27	0001	1011	1B [ESC]	59	0011	1011	3B ;	91	0101	1011	5B [123	0111	1011	7B {
28	0001	1100	1C [FS]	60	0011	1100	3C <	92	0101	1100	5C \	124	0111	1100	7C
29	0001	1101	1D [GS]	61	0011	1101	3D =	93	0101	1101	5D]	125	0111	1101	7D }
30	0001	1110	1E [RS]	62	0011	1110	3E >	94	0101	1110	5E ^	126	0111	1110	7E ~
31	0001	1111	1F [US]	63	0011	1111	3F ?	95	0101	1111	5F _	127	0111	1111	7F [DEL]

ASCII

- First created in 1963
 - American Standards Institute
 - (in reality IBM and AT&T)
 - The ASCII-67 version is the one that stuck
 - First one with lowercase characters
- 7 bits per character
 - 128 characters

Codepages

- Started around 1985
 - Used the 8th bit
 - Codepages for different languages
 - 128 characters match ASCII
 - OEM (IBM PC / DOS)
 - ANSI (Windows)
 - ISO-8859-*

Unicode

- Unicode Consortium started in 1991
 - Unicode 1.0
 - 16-bit (UCS-2)
 - Unicode 2.0
 - 21-bit (UTF-8, UTF-16, UTF-32)
 - UTF-8 & UTF16 are variable length encodings



Unicode makes things easier?

- Different encodings
 - UCS-2
 - UTF-8
 - UTF-16
 - UTF-16BE
 - UTF-32
- Characters have different sizes in memory
 - Also with UTF-16, even with UTF-32
 - Exception is UCS-2 which is antiquated technology
 - Complicates string functions
- Unicode is pretty much just the set of characters



Windows

- Started with ANSI (8-bit)
 - Support for OEM codepages
- Moved to UCS-2 (16-bit)
 - New widestring API's (or double byte)
- Moved to UTF-16 (16-bit or more)
 - Changed their double byte API's

DataFlex today

- Strings in DataFlex are OEM
 - We never changed to ANSI
- Conversions are done for most Window API calls
- DataFlex uses the single byte Windows API's

Project NextGen

Codename: NextGen

- Work started 2,5 years ago
 - Planning started way before that
- Dedicated resources were hired for the project

- Goal is to make DataFlex:
 - **Fully Unicode**
 - **64-bit capable**

DataFlex strings will be UTF-8

- Why?
 - UTF-8 provides backwards compatibility
 - First 128 characters match ASCII
 - The web is already UTF-8
 - UTF-8 is the best encoding
 - SQL Server 2019 will support UTF-8
- Microsoft is experimenting with UTF-8 as default 'code-page'

Source will become UTF-8

- Source files will be stored as UTF-8
 - Likely with a BOM, so the compiler can recognize old files that are still OEM
- Literals will support UTF-8
- Names will only support ASCII characters

Demo...



Database in NextGen

- SQL will be the expected database
- Possibilities for the DataFlex Embedded Database:
 - A Unicode version of the Embedded Database
 - No backwards compatibility
 - An OEM to Unicode data conversion utility would be provided
 - Continued support as OEM (without Unicode)
 - Collating sequence will be a challenge
 - Likely still incompatible with previous (32-bit OEM) Embedded Database
 - No embedded database at all

What code changes are needed?

- Convert to the wide character windows API's
 - Since Windows is UTF-16 a conversion will be needed
 - Not using the W functions will work in a lot of cases, but non ASCII characters will display incorrectly
- Remove / change all conversions
 - ToOEM / ToANSI indicates that something needs to change
- Check string manipulations
 - Bytes do not equal code points any more
- Not a line of code changed in Order Entry

Status

- Most native components are converted
 - Runtime, compiler, CDS
- Studio is mostly converted
- String functions are being implemented right now
- Open items
 - Embedded Database
 - Connectivity Kits
 - Tools (Database Explorer, Database Builder, ...)
- Progress is steady...



64-bit

What is 64-bit?

- The registry size of the processor
- Length of a memory address
- Since 1995 we used 32-bits
 - Addresses up to 4GB of memory
- Since 2001 64-bit versions of windows were available
 - Can address a lot more gigabytes of memory..
 - Not everyone moved to 64-bit directly
 - 64-bit windows can run 32-bit software



Why do I need 64-bit?

- Because the world is moving towards 64-bit
- Communicate with other 64-bit software
- To be more competitive

64-bit capable

- You will choose per project between 64-bit and 32-bit
- We expect 32-bit to be around for a while
 - A 64-bit application cannot use 32-bit DLLs
 - This includes COM components
 - All third party components you use need to be 64-bit
 - You need time to migrate your code
- New projects should be 64-bit by default

Language changes

- New LongPtr type
 - Integer type that is the same size of a pointer
 - 32-bit on 32-bit and 64-bit on 64-bit
- Integer stays 32-bit
- Pointer is now an Address
 - Used to be an integer
- Handle becomes LongPtr
- New compiler switch

```
#REPLACE Pointer Integer  
#REPLACE Handle Integer
```



```
#REPLACE Pointer Address  
#REPLACE Handle LongPtr
```

```
#IFDEF IS$WIN64
```

```
#ELSE
```

```
#ENDIF
```

Package changes

- Various Integer to LongPtr changes
 - External API's
 - Window messages
- Several Integer to Address changes
 - Invalid usage of Integer
 - Bad habits since the beginning of DataFlex 😊

Internal changes

- Lots of changes in the C codebase of the runtime
 - Pointers were passed as integer a lot...
- Multiple expression evaluator changes
 - Caused by the new LongPtr type
- Brand new linker
 - The part of the compiler producing the executable
 - Already in 19.1 (embed manifest files!)
- Converted all dependencies

Demo..



Converting your application...

- All your third party dependencies need to be 64-bit
 - All DLL's / COM controls
 - COM API's might be slightly different on 64-bit
- Changes *might* be needed in your code
 - No pointers values in integers any more!
 - External API's might require the use of LongPtr
 - Most of these are in more low level code
- Not a line of code changed in Order Entry
- With 19.1 we start helping you prepare
 - See Stephen's "Getting your applications ready for DataFlex NextGen"

The NextGen environment..

- All tools will be 64-bit (Studio, DB Explorer, ..)
 - Also builds and debugs 32-bit applications
- WebApp Server will be 64-bit
 - Also runs 32-bit applications
- Client installer will both 32-bit and 64-bit
 - Will work on 32-bit only machines

Status

- Most of the work is done
- We have a pretty stable environment
- Lots of testing is being done
 - Most current work comes out of test results



Moving forward..

The current DataFlex..

- Will be continued to be supported for a while
 - Based on the current codebase
 - New features will be backported
 - So for a while there will be 2 versions DataFlex
- Gives the new DataFlex time to mature
 - Experience with converting will grow in the community
- Gives you more time to migrate

When?

- After the 19.1 the focus of the entire team will shift to NextGen
 - It will become our default platform for developing new features
 - The entire team will start testing and reporting issues
- First technical previews should become available later this year..
- We are shooting for a first release the first quarter of 2020..



Thank you for your time!

I'll be around for any questions you might have...