디지털 데이터 표현

JK Kim

@pr0neer

forensic-proof.com

proneer@gmail.com

개요

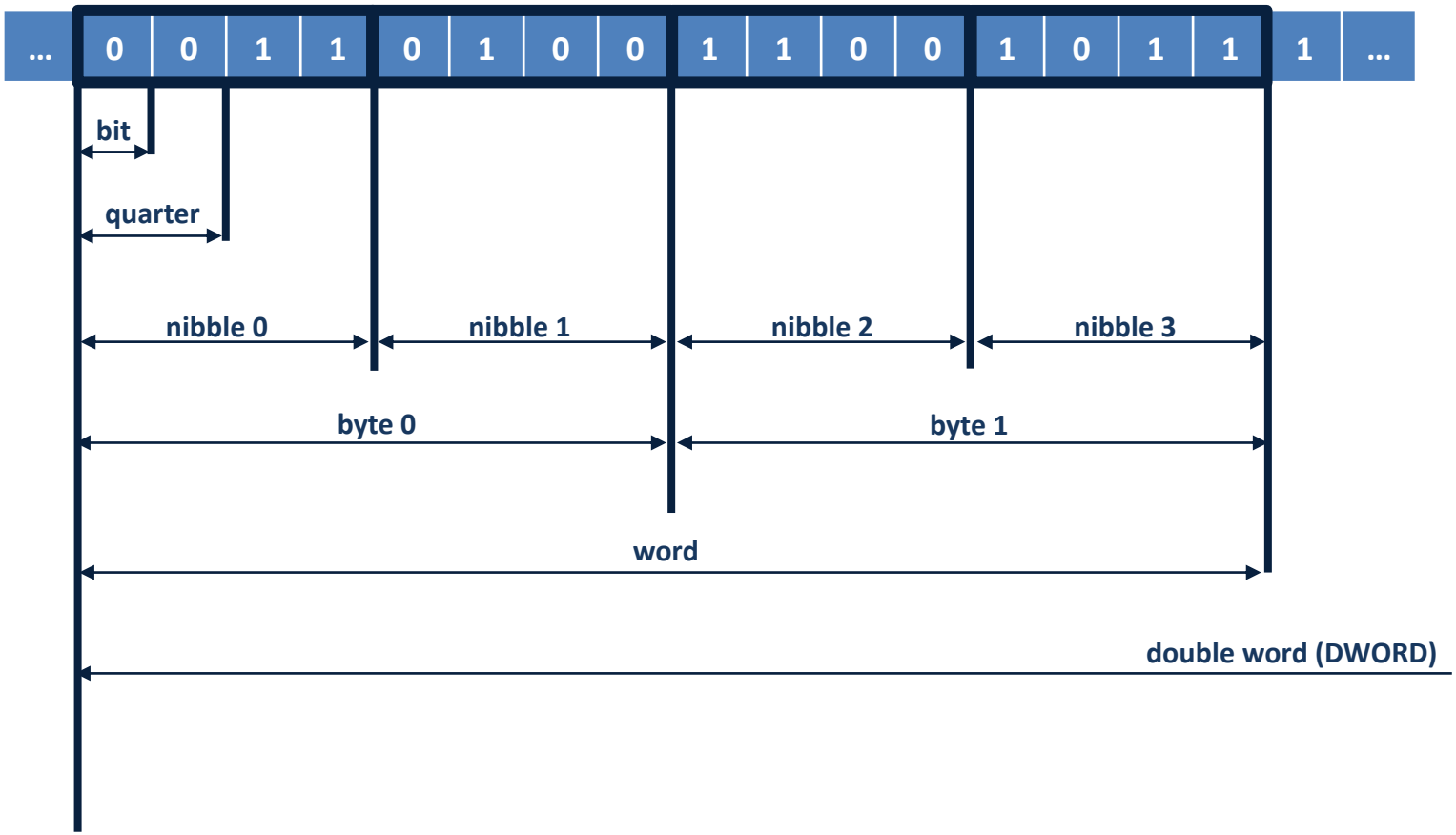
1. 데이터 표현 방식
2. 수 표현 방식
3. 문자 표현 방식
4. 시간정보 표현 방식

데이터 표현 방식

Security is a people problem…

데이터 표현 방식

물리적 데이터 단위



데이터 표현 방식

물리적 데이터 단위

단위	크기	
Kilobyte (KB)	1,024 bytes	2^{10} bytes
Megabyte (MB)	1,024 KB	2^{20} bytes
Gigabyte (GB)	1,024 MB	2^{30} bytes
Terabyte (TB)	1,024 GB	2^{40} bytes
Petabyte (PB)	1,024 TB	2^{50} bytes
Exabyte (EB)	1,024 PB	2^{60} bytes
Zettabyte (ZB)	1,024 EB	2^{70} bytes
Yottabyte (YB)	1,024 ZB	2^{80} bytes

데이터 표현 방식

논리적 데이터 단위

단위	설명
필드 (field)	• 여러 바이트 혹은 워드가 모여서 구성 • 논리적 처리의 최소 단위
레코드 (record)	• 프로그램의 자료 처리 기본 단위
블록 (block)	• 저장매체 입출력의 기본 단위
파일 (file)	• 연관된 레코드의 집합으로 응용프로그램의 처리 단위
데이터베이스 (database)	• 계층적 구조를 갖는 레코드의 집합 • 접근, 수정, 관리가 용이하게 잘 조직된 정보의 묶음

데이터 표현 방식

엔디언 (Endian)

- **컴퓨터 저장공간의 바이트 배열 순서**

종류	0x12345678 표현	장점	적용 대상
빅 엔디언 (Big-endian)	12 34 56 78	직관적으로 디버깅 용이	IBM, Motorola, Sparc, **Network**, …
리틀 엔디언 (Little-endian)	78 56 34 12	하위 바이트 접근이 용이	INTEL(x86), …

- **바이 엔디언 (Bi-endian)**
  - 빅 엔디언과 리틀 엔디언 중 하나를 선택할 수 있도록 설계
  - ARM, PowerPC, DEC Alpha, MIPS, IA-64 등

수 표현 방식

Security is a people problem…

수 표현 방식

수 체계

- 실생활에서는 10진수 사용
- 디지털 데이터는 2진수를 사용

수 체계	표현 범위
2진수 (binary)	0, 1
8진수 (octal)	0 – 7 (0, 1, 2, 3, 4, 5, 6, 7)
10진수 (decimal)	0 – 9 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
16진수 (Hexadecimal)	0 – 9, A – F (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

수 표현 방식

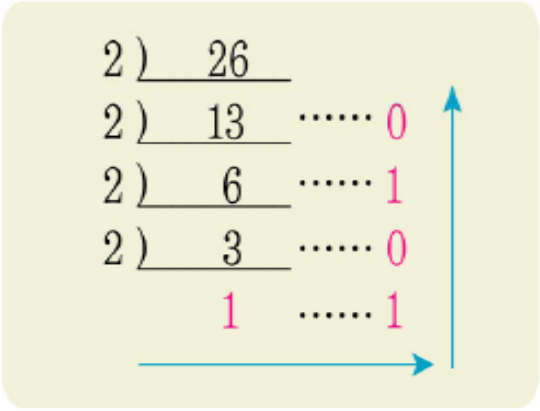
바이트 표현

- **1바이트(8비트) 2진수 표현**

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

- **26_{10}의 2진수 표현**

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	0	0	1	1	0	1	0



수 표현 방식

바이트 표현

- **2진수 계산**

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	0	0	1	1	0	1	0

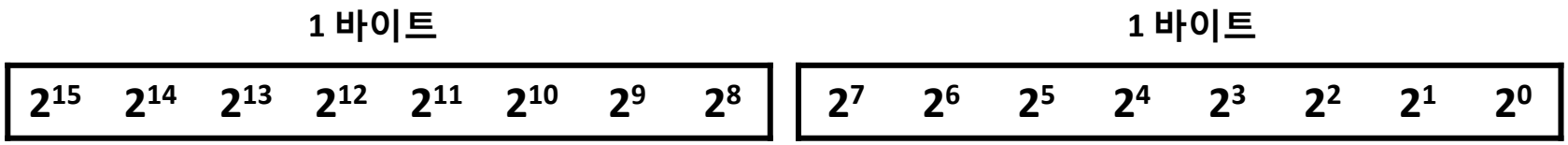
  - 각 비트와 2의 자승의 곱으로 계산
    - ✓ (**0** X 2^7) + (**0** X 2^6) + (**0** X 2^5) + (**1** X 2^4) + (**1** X 2^3) + (**0** X 2^2) + (**1** X 2^1) + (**0** X 2^0) = **26**

- **최소값**: 00000000 (0)
- **최대값**: 11111111 (255)
- 256개의 값 표현 (0 ~ 255)
- 255가 넘는 수를 표현하기 위해서는 추가적인 바이트 필요

수 표현 방식

바이트 표현

- **2바이트(16비트)**



  - **최소값**: 00000000 00000000 (0)
  - **최대값**: 11111111 11111111 (65535)

수 표현 방식

바이트 표현

- **16진수의 필요성**
  - 디지털 데이터는 실제 2진수를 사용하지만, 표기하기에는 부적절
  - 직관적인 표기를 위해 니블(4바이트) 단위로 묶어서 16진수로 표현

2진수	16진수
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

2진수	16진수
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

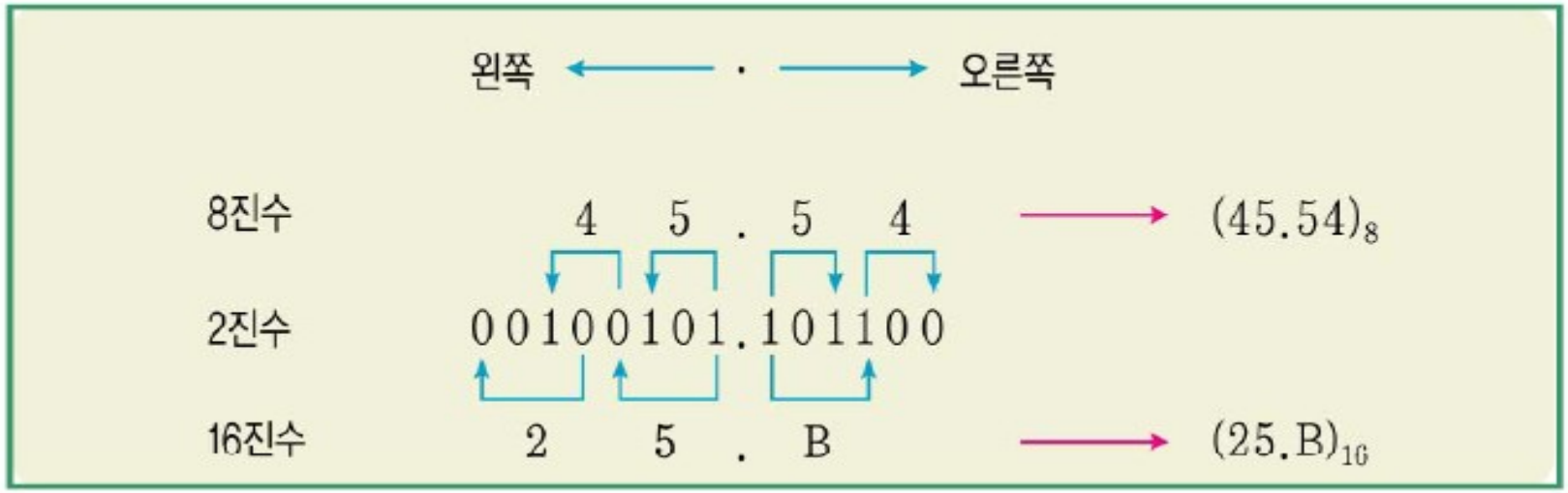
수 표현 방식

수 변환

- **2진수 ➔ 16진수 변환**

0	1	1	0	0	1	1	1	1	0	1	0	1	1	0	0
6				7				A				C			

- **37.11_{10}의 2/8/16진수 상호간의 변환**



수 표현 방식

음수 표현

- **음수 표현 방법**
  - **부호화-크기 표현** (signed-magnitude representation)
  - **1의 보수 표현** (1's complement representation)
  - **2의 보수 표현** (2's complement representation)

수 표현 방식

음수 표현 – 부호화 크기 표현

- **최상위 비트(MSB)를 부호 비트로 사용**
  - 표현 가능 범위 : $-(2^{n-1}-1) \sim +(2^{n-1}-1)$
  - 8비트 데이터 표현 범위 : $-127(2^7 - 1) \sim +127(2^7 - 1)$ **(11111111 ~ 011111111)**

- **한계**
  - 수 계산 시 부호 비트와 크기 부분을 별도로 처리
  - 0에 대한 표현이 중복
    - ✓ 양수 0 : 0 0000000
    - ✓ 음수 0 : 1 0000000

수 표현 방식

음수 표현 – 1의 보수와 2의 보수 표현

- **1의 보수 표현 – 양수의 모든 비트를 반전시켜 음수 표현**
  - 표현 가능 범위 : $-(2^{n-1}-1) \sim +(2^{n-1}-1)$
  - 8비트 데이터 표현 범위 : $-127(2^7 - 1) \sim +127(2^7 - 1)$ **(11111111 ~ 10000000)**
  - 여전히 0에 대한 표현이 중복

- **2의 보수 표현 – 1의 보수 결과에 1 더하기 (최상위 비트가 올림 수 발생 시 버림)**
  - 표현 가능 범위 : $-2^{n-1} \sim +(2^{n-1}-1)$
  - $7(2^7 - 1)$

+11 = 0 0001011	**+36** = 0 0100100
- 11 = 1 1110100 (1의 보수)	**- 36** = 1 1011011 (1의 보수)
- 11 = 1 1110101 (2의 보수)	**- 36** = 1 1011100 (2의 보수)

수 표현 방식

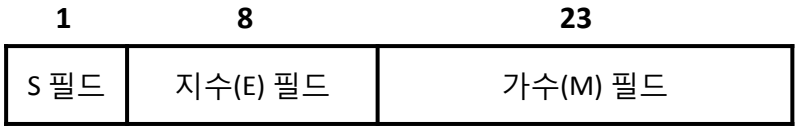
음수 표현 가능 범위와 비트 패턴

10진수	부호화 크기 표현	1의 보수 표현	2의 보수 표현
127	01111111	01111111	01111111
126	01111110	01111110	01111110
:	:	:	:
1	00000001	00000001	00000001
+0	00000000	00000000	00000000
-0	10000000	11111111	-
-1	10000001	11111110	11111111
-2	10000010	11111101	11111110
:	:	:	11111101
-126	11111110	10000001	:
-127	11111111	10000000	10000001
-128	-	-	10000000

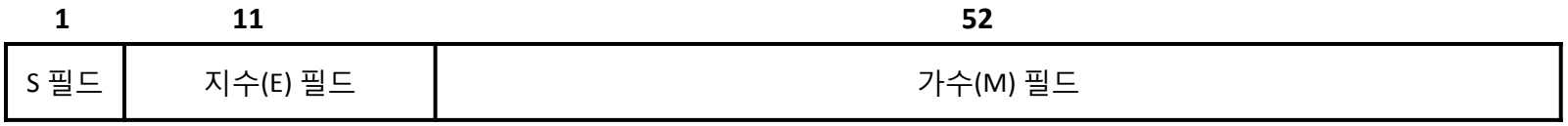
수 표현 방식

부동소수점 표현 (Floating-Point Representation)

- **부동소수점 표현의 국제 표준(IEEE 754)**
  - 표기법, 지수와 가수를 구분 : ± 1.1bbbb…bbb x 2±E
- **±단일 정밀도 (32비트) 형식**
  - 표현 영역 : 10^{-38} ~ 10^{38}



- **복수 정밀도 (64비트) 형식**
  - 표현 영역 : 10^{-308} ~ 10^{308}



부동소수점 표현 (Floating-Point Representation)

- **$(-13.625)_{10}$의 IEEE 754 표현**

1. 13.625를 2진수로 바꾼 후, 표준형으로 변환

   ✓ $(13.625)_{10} = (1101.101)_2 = 1.101101 \times 2^3$

2. 단일 정밀도 형식으로 표현

   ✓ 부호(S) 비트 = 1(–)

   ✓ 지수 E = 00000011 + 01111111 = 10000010 (+ bias 127)

   ✓ 가수 M = 10110100000000000000000 (소수점 좌측 1을 제외)

S 필드(1)	지수(E) 필드(8)	가수(M) 필드(23)
1	10000010	1011 0100 0000 0000 0000 000

문자 표현 방식

Security is a people problem…

문자 표현 방식

아스키 (ASCII) 코드

- **미국표준협회(ANSI, American National Standards Institute)가 제정한 상호 정보 교환 코드**
  - 1비트 패리티와 7비트(존비트 3개, 디지트 4개)를 이용하여 128개(2^7)의 문자 구성
    - ✓ 디지트(digit) 비트 : 4개
    - ✓ 출력 불가능한 문자 : 33개
    - ✓ 출력 가능한 문자 : 95개 (영문알파벳 대소문자:52, 숫자:10, 특수문자:32, 공백문자)
  - 이후 1비트가 추가되어 256개(2^8) 문자를 표현하는 확장(Extended) ASCII로 변화
  - 아스키 코드 앞에 비트 0을 넣어 8비트 인코딩으로 확장 ➔ ISO 8859

문자 표현 방식

아스키 (ASCII) 코드

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

문자 표현 방식

확장 ASCII, ISO/IEC 8859

- **확장 ASCII (패리티 비트 ➔ 문자 비트)**
  - 8비트 : 문자 비트, 256(2^8) 문자 표현 (기존 ASCII 코드에 127개 문자 추가)
- **ISO/IEC 8859**
  - 127개의 추가 문자를 효과적으로 활용하고자 다양한 문자 집합 사용

문자 집합	설명
ISO/IEC 8859-1	라틴-1 서유럽
ISO/IEC 8859-2	라틴-2 중앙유럽
ISO/IEC 8859-3	라틴-3 남유럽
ISO/IEC 8859-4	라틴-4 북유럽
ISO/IEC 8859-5	라틴/키릴
ISO/IEC 8859-6	라틴/아랍
ISO/IEC 8859-7	라틴/그리스
ISO/IEC 8859-8	라틴/히브리

문자 집합	설명
ISO/IEC 8859-9	라틴-5 터키
ISO/IEC 8859-10	라틴-6 노르딕
ISO/IEC 8859-11	라틴/타이
ISO/IEC 8859-12	라틴/데바나가리
ISO/IEC 8859-13	라틴-7 발트해 연안
ISO/IEC 8859-14	라틴-8 켈트
ISO/IEC 8859-15	라틴-9
ISO/IEC 8859-16	라틴-10 남동유럽

문자 표현 방식

BCD 코드

- **Binary-Coded Decimal Code**
  - 하나의 문자를 7비트로 표시
  - 1비트 패리티, 6비트 문자 (존비트 2개, 디지트 4개)

1	2	4
패리티	존(zone)	디지트(digit)

  - **존(zone) 비트**
    - ✓ 00 : 숫자 0~9로 이루어짐
    - ✓ 01 : + 기호와 대문자 (A~I) 등으로 이루어짐
    - ✓ 10 : - 기호와 대문자 (J~R) 등으로 이루어짐
    - ✓ 11 : / 기호와 대문자 (S-Z) 등으로 이루어짐

문자 표현 방식

BCD 코드

- **Binary-Coded Decimal Code**

	00 (0)	01 (1)	10 (2)	11 (3)
0000 (0)		+	-	공백처리
0001 (1)	1	A	J	/
0010 (2)	2	B	K	S
0011 (3)	3	C	L	T
0100 (4)	4	D	M	U
0101 (5)	5	E	N	V
0110 (6)	6	F	O	W
0111 (7)	7	G	P	X
1000 (8)	8	H	Q	Y
1001 (9)	9	I	R	Z
1010 (A)	0			
1011 (B)	=	.	$	
1100 (C)	,	)	*	(
1101 (D)				
1110 (E)				
1111 (F)				

문자 표현 방식

EBCDIC 코드

- **Extended Binary-Coded Decimal Interchange Code**
  - IBM에서 대형 운영체제에 사용하기 위해 확장한 코드
  - IBM S/390 서버의 운영체제인 OS/3990에서 사용되던 텍스트 파일용 코드
  - 이전 응용프로그램 및 데이터베이스와 호환을 위해 아직도 사용
  - IBM PC 운영체제는 ASCII 코드 사용

문자 표현 방식

한글 표현 방식

- **한글 조합형**
  - 1980년대 초반 ~ 1990년대 중반까지 널리 사용
  - 현대 한글 음절 11,172개 모두를 표현할 수 있는 방식
  - 초성, 중성, 종성을 각각 별도로 처리하여 모든 글자를 조합
  - n바이트 조합형 : 최소 2바이트 ~ 최대 5바이트 (가변적)
  - 3바이트 조합형 : 초성(1바이트), 중성(1바이트), 종성(1바이트), 채움 문자 ➔ 4바이트
  - 2바이트 조합형 : 최상위 설정 1비트, 초성 5비트, 중성 5비트, 종성 5비트
    - ✓ ASCII와 겹치지 않도록 최상위 비트는 1로 설정
    - ✓ 상용 조합형, 삼성 조합형, 금성 조합형, 도깨비 조합형 등
    - ✓ 상용 조합형이 최종까지 살아남음

문자 표현 방식

한글 표현 방식

- **한글 완성형**
  - 한글 구조와 관계 없이 코드를 배당하여 표현
  - 7비트 완성형
    - ✓ 청계천 주변 상가에서 만들어져 보급 ➔ 청계천 한글
    - ✓ MSB를 사용할 수 없는 영문 프로그램에서 한글 표현을 위해 생성
    - ✓ 표현 가능한 글자 : 1300 여자
    - ✓ 영문이 한글로 표현되는 오류도 존재 (dBASE ➔ 늦ASE)
  - 2바이트 완성형
    - ✓ 기존 문자 체계와의 호환성과 국내 업체의 형평성을 위해 새로운 완성형 제정
    - ✓ 현재 **KS X 1001**로 표준화

문자 표현 방식

한글 표현 방식

- **한글 완성형 – KS X 1001(KSC5601-1987)**
  - 한국 산업 규격의 한국어 문자 집합으로 정식명은 “정보 교환용 부호계 (한글 및 한자)”
  - 1974년 처음 제정, 현재는 2004년 개정된 “KS X 1001:2004” 사용
  - 표현 영역 : 0x2121 – 0x7E7E (8,836 문자)
    - ✓ 0x21 – 0x2C : 특수 문자, 한글 낱자, 괘선 조각, 외국 문자 (히라가나, 가타카나, 그리스, 키릴 등)
    - ✓ 0x30 – 0x48 : 한글 글자 마디 영역 (자주 사용하는 한글 2,350자)
    - ✓ 0x49 : 사용자 정의 영역 A
    - ✓ 0x4A – 0X7D : 한자 영역
    - ✓ 0X7E : 사용자 정의 영역 B
  - 부속서 3를 통해 상용 조합형 반영, 한글 채움 문자를 통해 표현할 수 없는 문자 추가
    - ✓ (채움) ㅆ ㅠ (채움), 한글 채움 문자(24 54, EUC-KR에서는 A4 D4)

문자 표현 방식

한글 표현 방식

- **한글 완성형 인코딩 – EUC-KR**
  - AT&T에서 만든 아시아계 문자를 표현하기 위한 확장 유닉스 코드 (Extended Unix Code)
    - ✓ 확장 유닉스 코드는 한국어, 중국어, 일본어에 주로 사용되는 8비트 문자 인코딩 방식
    - ✓ EUC-KR, EUC-CN, EUC-TW, EUC-JP 등
  - KS X 1001과 ASCII를 통합하여 제정
    - ✓ \에 해당하는 문자는
    - ✓ 0 – 127 : KS X 1003 배당 (ASCII 포함)
    - ✓ 128 – 255 : KS X 1001 배당, 행과 열에 128을 더한 코드값(한 바이트) ➔ 2바이트 표현
  - KS X 1001의 단점인 일부 문자 표현이 불가능하다는 단점을 그대로 지님

문자 표현 방식

한글 표현 방식

- **한글 완성형 인코딩 – CP949 인코딩**
  - EUC-KR에서 포함하지 못하는 문자를 표현하기 위해 마이크로소프트에서 제정
  - 인터넷 정보 교환의 표준은 아님
  - KS X 1001(KSC5601-1987)
    - ✓ 2바이트 문자 코드로 한글 2350자 표현
  - CP949
    - ✓ KSC5601-1987 + 8822 글자 = 11172 글자 표현
  - KS X 1001은 한글 채움 문자를 사용하여 모든 현대 한글을 표현할 수 있지만,
  - 현재 대부분 EUC-KR을 사용하지 않고 CP949를 사용하여 한글 표현

문자 표현 방식

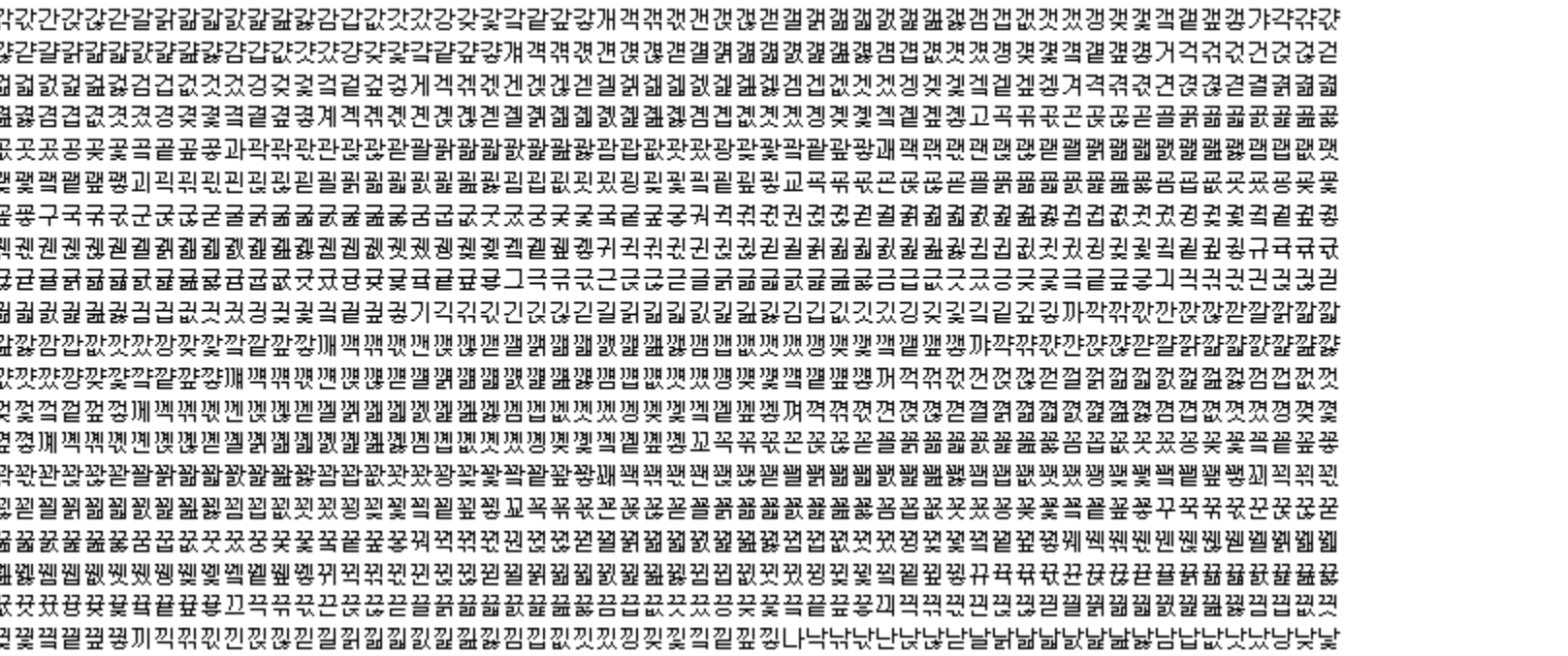
한글 표현 방식

- **유니코드**
  - 전 세계 모든 문자를 일관되게 표현하기 위한 산업 표준
  - ISO 10646으로 정의된 UCS(Universal Character Set)
  - UCS는 110만개 이상의 코드가 있지만, 첫 65536만 사용
    - ✓ BMP : Basic Multilingual Plane; 기본 다국어 평면
    - ✓ BMP 영역 이외는 고대 이집트 상형 문자나 쓰임이 적은 한자를 표현
    - ✓ BMP 영역도 서로 다른 인코딩이나 미래 확장성을 고려하여 할당하지 않은 영역 존재
  - 1991년 유니코드 1.0, 2009년 10월 유니코드 5.2 발표
  - 한글은 1996년 유니코드 2.0부터 11,172자 포함 (유니코드의 17% 점유)
    - ✓ **[1010 1100 0000 0000]** [0xAC00](44,032)번 [가] 부터
    - ✓ **[1101 0111 1010 1111]** [0xD7AF](55,215)번 [힣] 까지

문자 표현 방식

한글 표현 방식

- **유니코드**
  - 초성에 중성과 종성을 붙여 비슷한 모양이 반복
  - 한글의 과학성으로 문자 코드를 초성 [i], 중성 [j], 종성 [k] 변수를 사용하여 표현 가능
    - ✓ **[0xAC00 + (28*21*i) + (28*j) + k]**



문자 표현 방식

한글 표현 방식

- **유니코드 인코딩 – UCS-2, UCS-4**
  - UCS-2
    - ✓ 각 글자를 0 – 65535 (0xFFFF) 사의의 코드값으로 매핑하여 인코딩
    - ✓ UCS-2는 BMP 영역 밖은 표현이 불가능
    - ✓ UCS-2를 BMP 영역도 표시가 가능하도록 확장 ➔ UTF-16
  - UCS-4
    - ✓ 4바이트 (0xFFFFFFFF)로 한 글자를 표현
    - ✓ UCS-2보다 더 많은 글자 표현

문자 표현 방식

한글 표현 방식

- **유니코드 인코딩 – UTF-8, 16 (UCS/Unicode Transformation Format)**
  - UTF-8
    - ✓ 유니코드 가변 길이 문자 인코딩 방식
    - ✓ 유니코드 한 문자 표현을 위해 1 – 4 바이트 사용
  - UTF-16
    - ✓ BMP에 속하는 문자들은 16비트로 인코딩, BMP 밖의 문자는 32비트로 인코딩

코드 범위 (16진수)	UTF-8 (2진수)	UTF-16 (2진수)
000000 – 00007F (ASCII 범위)	0xxxxxxx	00000000 0xxxxxxx
000080 – 0007FF	110xxxxx 10xxxxxx	00000xxx xxxxxxxx
000800 – 00FFFF	1110xxxx 10xxxxxx 10xxxxxx	xxxxxxxx xxxxxxxx
010000 – 10FFFF	11110zzz 10zzxxxx 10xxxxxx 10xxxxxx	110110yy yyxxxxxx 110111xx xxxxxxxx

시간정보 표현 방식

Security is a people problem…

시간정보 표현 방식

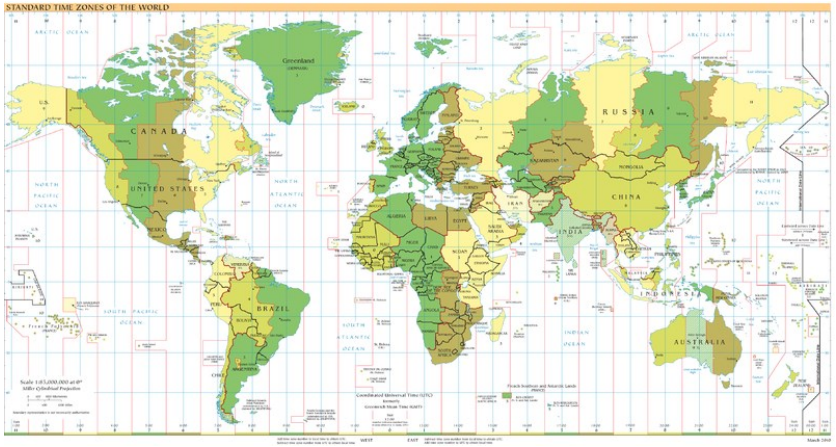
GMT (Greenwich Mean Time)

- **그리니치 표준시**
  - 영국 런던 외각의 그리니치 천문대(경도 0º)를 기준으로 한 평균태양시
  - 1925년 2월 5일부터 거의 반세기 동안 (~ 1972년 1월 1일)세계 표준시로 사용
  - 태양이 천문대의 남중 자오선을 통과하는 시각을 정밀하게 측정 (정오를 기준)
  - 서울은 동경 127 º 표준 자오선을 사용, 시간은 동경 135 º 표준 자오선(일본 표준시 JST)을 사용

시간정보 표현 방식

UTC (Universal Time, Coordinated)

- **협정 세계시**
  - 1972년 1월 1일 세계 표준시로 규정 (지구 자전 주기의 변화)
  - 원자 시계를 기준으로 한 국제 원자시(TAI: International Atomic Times)를 기준으로 한 시간
  - 국제 원자시는 세슘 원자의 진동수에 기반하여 측정
  - 협정 세계시의 하루는 보통 86, 600 SI 초, 실제 태양시는 86,400 초보다 약간 더 김
  - 하루의 제일 마지막 1분을 61초로 계산하는 윤초(leap second)를 사용
  - 현재 일반적으로 GMT = UTC 간주
  - 2008년 7월 시간 독립의 의미에서 법안 발의



시간정보 표현 방식

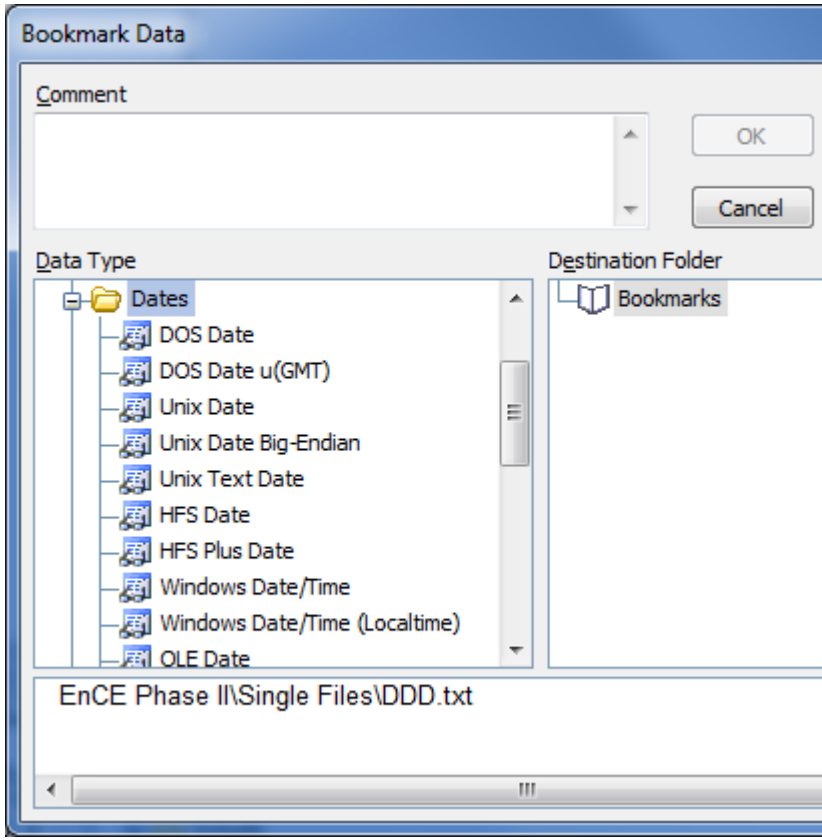
Local Time

- **현지 시간**
  - 수상 대상물이 위치한 지역의 현지 시간
  - 서울의 경우 UTC+9
  - 파일 속성의 경우
    - ✓ FAT : 4바이트 크기의 현지 시간
    - ✓ NTFS : 8바이트 크기의 GMT 시간
  - 정확한 수사를 위해 수사 대상물의 시간 정보를 UTC ➔ 현지 시간으로 변환 필요

시간정보 표현 방식

다양한 시간 형식

- DOS Date
- DOS Date u(GMT)
- Unix Date
- Unix Date Big-Endian
- Unix Text Date
- HFS Date
- HFS + Date
- Windows Date/Time
- Windows Date/Time (Localtime)
- OLE Date
- Lotus Date



시간정보 표현 방식

DOS Date/Time

- MS-DOS에서 사용된 시간 저장 형식
- 현재 날짜와 시간(Local Time) 저장 (날짜 : 2바이트, 시간 : 2바이트)
- 표현 범위
  - 시작 : 1980년 01월 01일 00:00:00 (00 : 21 : 00 : 00)
  - 끝 : 2107년 12월 31일 23:59:58 (FF : 9F : BF : 7F)

시간정보 표현 방식

DOS Date/Time

- **구조**

날짜 (H)								날짜 (L)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Y	Y	Y	Y	Y	Y	Y	M	M	M	M	D	D	D	D	D

시간 (H)								시간 (L)							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
h	h	h	h	h	m	m	m	m	m	m	s	s	s	s	s

- Y : 1980년을 기점으로 한 연도
- M : 1 – 12
- D : 1 – 31
- h : 0 – 23
- m : 0 – 59
- s : 0 – 29 (2초 단위)

시간정보 표현 방식

UNIX Date/Time

- 1970년 1월 1일 0시(UTC)를 기준으로 1초씩 증가해온 시간
- (1234567890)유닉스 시간
  - 2009년 2월 13일 금요일 23시 31분 30초 (국내 2009년 2월 14일 (발렌타인) 08시 31분 30초)
- 4바이트(32비트)의 정수값으로 표현 ➔ 2^{32} = 4,294,967,296
- 표현 범위
  - 시작 : 1970년 01월 01일 00:00:00 (UTC)
  - 끝 : 2038년 01월 19일 03:14:07 (UTC)
- 유닉스 시간 표현을 위해 time_t 자료형사용
- 표현 범위 문제 해결을 위해 64비트 아키텍쳐를 지원하는 운영체제는 time64_t 사용

시간정보 표현 방식

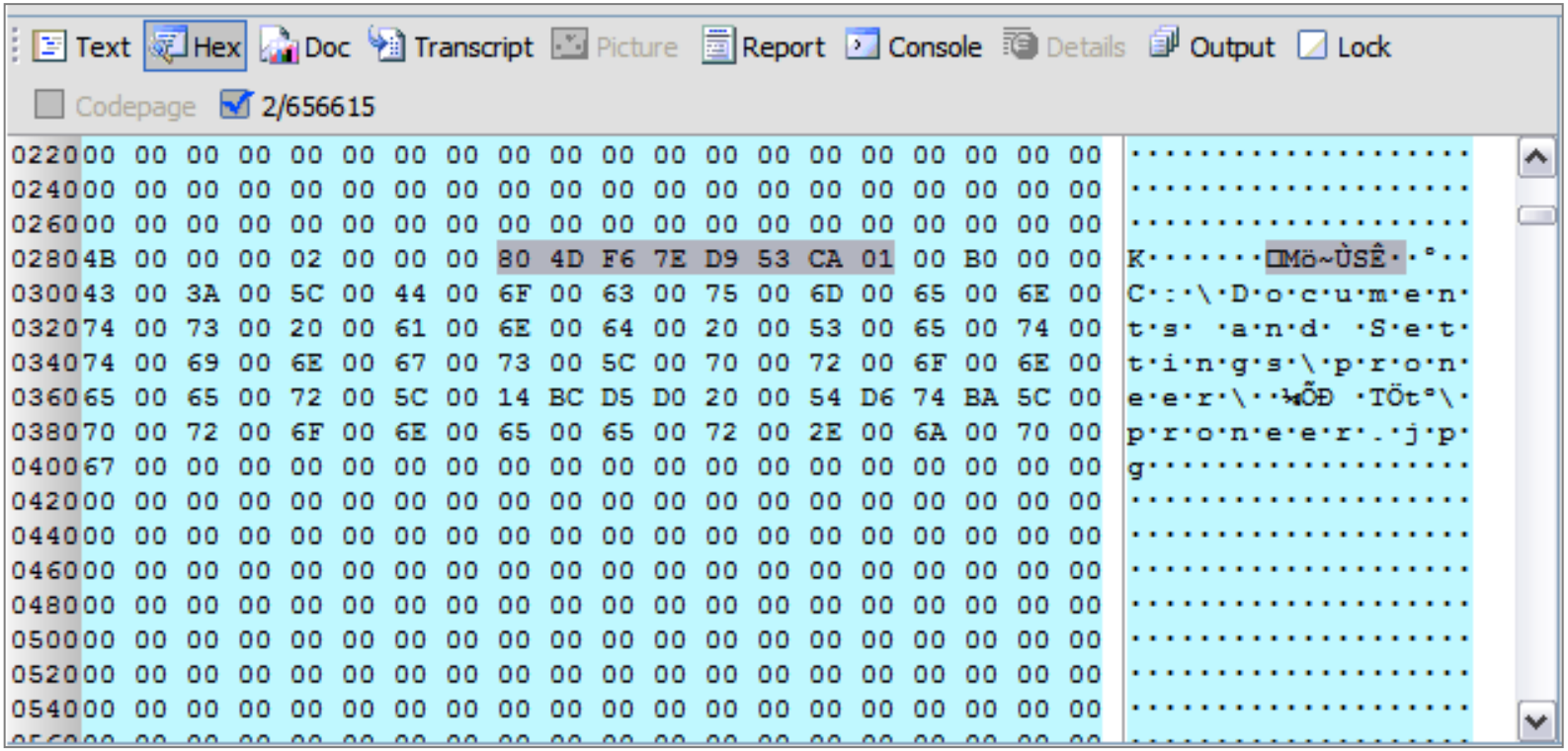
WINDOWS 64-Bit Time Stamp

- 윈도우에서 사용하는 8바이트(64비트) 시간 형식
  - 2^{64} = 18,446,744,073,709,500,000
- 1601년 01월 01일 자정 (UTC)를 기준으로 100나노초 간격으로 증가
- 표현 범위 (날짜 : 4바이트, 시간 : 4바이트)
  - 시작 : 01601년 01월 01일 00:00:00 (00000000 : 00000000)
  - 끝 : 60056년 03월 28일 14:36:10 (FFFFFFFF : FFFFFFFF)
- 시간 표현을 위해 FILETIME 구조체 사용
- FILETIME to time_t
  - time_t = (FILETIME – 0x19DB1DED53E8000) / 10000000

시간정보 표현 방식

WINDOWS 64-Bit Time Stamp

- 8 바이트의 최상위 바이트는 01h



시간정보 표현 방식

MOBILE Date/Time

형식	저장 값(16진수)	시간 표현
10진수	08 02 24 15 46 02	2008년 02월 24일 15시 46분 02초
16진수 (Little-Endian)	D8 07 01 13 13 15 33 (0x07D8 = 2008)	2008년 01월 19일 19시 21분 51초
16진수	07 03 02 0C 33 36 (0x07 = 2007)	2007년 03월 02일 12시 51분 54초
16진수 (Big-Endian)	07 D9 08 05 09 15 06 (0x07D9 = 2009)	2009년 08월 05일 09시 21분 06초
ASCII	32303036 2E 3034 2E 3035 20 3135 3A 3031	2006년 04월 05일 15시 01분
	3036 2E 3034 2E 3035 20 3135 3A 3031	2006년 04월 05일 15시 01분
Qualcomm TimeStamp	0x054BEC34 (1980년 1월 6일 오전 12시 00분 00초 기준)	2008년 02월 24일 15시 08분 53초
	0x383936343630313139 (896460119초, ASCII)	2008년 06월 02일 16시 41분 59초
Anycall TimeStamp	0x93E18003 (1896년 1월 6일 오전 12시 00분 기준)	2007년 10월 07일 22시 35분

질문 및 답변